Ph.D. Dissertation

Numerical computing of extremely large values of the Riemann-Siegel Z-function

Author:

Norbert Tihanyi

Supervisor:

Attila Kovács, PhD.

Eötvös Loránd University

Doctoral School of Informatics

Numeric and Symbolic Computations Doctoral Program

Head: Prof. Erzsébet Csuhaj-Varjú, DSc.

Program Director: Prof. Ferenc Weisz, DSc.

May 2019

Acknowledgements

I would like to thank many people who inspired me during my doctoral studies.

First and foremost, I would like to thank my supervisor Attila Kovács for his continuous support and collaboration over the years.

Second, I would like to thank professors G.A. Hiary, and J. Pintz for their useful advice.

Third, I am very grateful to my family for supporting my doctoral studies. I would like to thank my mother, and my wife for their unconditional support and love.

Finally, I am very grateful for the opportunity of accessing the ATLAS Super Cluster operating at Eötvös Loránd University and for using the capacity of the SZTAKI Desktop Grid operated by the Laboratory of Parallel and Distributed Systems in the Institute for Computer Science and Control of the Hungarian Academy of Sciences.

Contents

Preface

Primes are the building blocks of mathematics and affect all areas of our lives. Primes will come up when we use our ATM card, we buy something on e-bay, or we send an email to our colleague. Primes are one of the most frequently discussed topics in mathematics, especially in number theory. The beauty of primes is their simultaneous simplicity and complexity. In the theory of prime numbers, sometimes it is not clear what really makes things easy or hard. A striking example is the factorization of integers. One can easily multiply two very large integers, however, factorizing such huge numbers can be tremendously hard.

"Given any two numbers, we may by a simple and infallible process obtain their product; but when a large number is given it is quite another matter to determine its factors. Can the reader say what two numbers multiplied together will produce the number 8,616,460,799? I think it unlikely that anyone but myself will ever know."- W. Stanley Jevons (1877)

Using a single desktop computer one can factor the aforementioned Jevons-number at glance: $8\,616\,460\,799 = 96\,079 \times 89\,681$. For large numbers, the problem is still challenging and even with the current state of the art supercomputing technology it could take millions of years to factor a 2048-bit integer. Can the reader say what two numbers multiplied together will produce the number

21924028190715002131790882556543405425287238544280259123791376833425848064815064829074140521682431845472805357734906904206020307105322037699825114176612832817898819338027698820573570210866400954667261677290754361638998912788376586893745691488151101985744368506577827823366565567917594911352356231126215560695423644037212438825028206394709799447706175173990923923021256442086350099238360387085370330429165137588348256341447849280411368673878814892176302655618429482826863876639912174421850652819605160967957410209986693044327161916673622289116287704772368230467651757662771697893606718231026787719553644167597254679211?[1]

[1] In 2019 this problem is far beyond our capacity.

In cryptography, the security of the RSA cryptosystem is based on the fact that integer factorization is computationally difficult. Many of the problems in number theory can be understood by laypersons, however solutions often require sophisticated mathematical background. The nature of the following two statements are very close to each other, but their proofs are "slightly" different:

- Any integer greater than 1 can be expressed as a product of prime numbers, which are unique up to order;
- Every even integer greater than 2 can be expressed as the sum of two primes.

The first statement is the Fundamental Theorem of Arithmetic which plays a central role in mathematics. The proof is straightforward and easy to understand. The second statement seems innocent enough not to cause any problems, however, the truth is quite the contrary. This is Goldbach's conjecture, one of the oldest problems in number theory. Despite extraordinary effort only particular solutions exist and the problem is still unsolved.

Integer factorization and Goldbach's conjecture are just a few examples where the distribution of prime numbers can be the key for success.

There is a deep connection between the distribution of prime numbers and the non-trivial zeros of the *Riemann zeta* function. The Riemann ζ function is an extremely important function of mathematics and physics. In mathematics, the number theoretic questions are the most interesting: what is the statistical distribution of the primes, how likely are two or more primes to be close together, how can we use this information to construct better encryption algorithms, etc.

In matrix theory it is well known that the zeros of ζ are related to the eigenvalues of certain random matrices, which are used in modelling energy levels of nuclei, financial markets, or big data. It has turned out that the non-trivial zeros can be considered as eigenvalues of an Hermitian operator [1]. In physics, the ζ function is applied in statistical mechanics, when bosonic or fermionic particles are encountered. On the other side of the tiny–large scale, the Riemann ζ can be applied in relativistic cosmology. E.g., the radius of the universe coincides with the absolute value of the Riemann zeta-function (with spherical geometry being assumed) hence infinitely many universe models exist [2].

The Riemann zeta is like an invisible spider-web, connecting the various scientific fields together.

The main goal of this thesis is to present an efficient searching method to find extremely large values of the Riemann zeta function on the critical line. Locating peak values of the zeta function is a promising method for getting a better understanding of the distribution of prime numbers. Solving multidimensional simultaneous Diophantine approximations we were able to create an efficient algorithm (called `RS-PEAK`) to find extremely large values of the Riemann zeta function.

During the last few decades many valuable results have been achieved regarding primes by applying huge amount of computing resources. For example, in 1985 the Mertens conjecture was disproved by Odlyzko and te Riele based on extensive computation of the zeros of the Riemann zeta function. Their method is an excellent example of a mathematical proof including a large amount of computational evidence.

The thesis is organized as follows. Chapter 1 contains a brief introduction to the Riemann zeta function, Riemann-Siegel Z-function (i.e $Z(t)$) and some related number theoretic functions. We present the previous results of other authors on finding peak values of the $Z(t)$ function.

In Chapter 2 we are focusing on the approximation of the main summand of the Riemann-Siegel Z function. We introduce the theory of continued fractions and the theory of Diophantine approximations. The main goal of this chapter is to present an efficient algorithm for solving n-dimensional Diophantine approximation problems.

In Chapter 3 we present the `RS-PEAK` algorithm that can be used to find candidates efficiently where large $Z(t)$ is likely. In this chapter we show how different areas of mathematics – such as Diophantine approximation and analytic number theory – can be used to achieve significant computational results in locating peak values of the Riemann zeta function.

The main result of this thesis is presented in Chapter 4. We present new computational results for finding extremely large values of $Z(t)$ applying the `RS-PEAK` algorithm. The computation environment was granted by the SZTAKI Desktop Grid operated by the Laboratory of Parallel and Distributed Systems at the Hungarian Academy of Sciences. Using SZTAKI Desktop Grid we found more than 5 million candidates where $Z(t) > 1000$ is likely. We present the largest known $Z(t)$ value. For $t = 3106788336290839656675405765936 82.05$ we have $Z(t) \approx 16874.202$.

To the best of our knowledge, at the time of writing this PhD thesis, this is the largest $|Z(t)|$ ever found.

The Riemann-Siegel Z-function has a very deep connection to the famous Riemann hypothesis. RH is equivalent to the statement that all local maxima of $Z(t)$ are positive

and all local minima are negative, and it has been suggested that if a counterexample exists then it should be in the neighbourhood of unusually large peaks of $Z(t)$. For this reason locating extremely large values of the Riemann zeta function on the critical line can reveal new behaviour of the distribution of prime numbers.

In Chapter 5 we present the possible future work and research directions.

This PhD thesis is based on the following core publications: [55], [56], [57], [58], [59], [60], [61].

Chapter 1

Introduction

It is well known that there are infinitely many primes, as proved by Euclid around 300 BC. Many results have been achieved in the past century with respect to prime numbers, but the distribution of them remained unknown. As of 2019 there is no known efficiently computable formula for generating prime numbers, and hunting for giant primes is still challenging. The largest known prime was discovered on December 7, 2018 by the Great Internet Mersenne Prime Search Project [3] and has 24 862 048 digits.

The main goal of this PhD thesis is to present an efficient searching algorithm called `RS-PEAK` to find extremely large candidates of the Riemann zeta function on the critical line. The Riemann zeta function has an interesting property: the distribution of its zeros has the same behaviour as the distribution of primes.

In this chapter we discuss the historical background of the Riemann zeta and related number-theoretic functions. We introduce the necessary definitions and notations and recall some earlier results to understand the connection between primes and the Riemann zeta function. The `RS-PEAK` algorithm will be presented in Chapter 3, however, at the end of this chapter we present the `RS-PEAK` algorithm in a nutshell to be able to reference it in the following chapters.

1.1 The Euler zeta function

The *Euler zeta function* was introduced by *Leonhard Euler* (1707-1783) in the first half of the eighteenth century. Euler's zeta function is defined for any real number $s > 1$ by the following sum

$$\zeta(s) = 1 + \frac{1}{2^s} + \frac{1}{3^s} + \frac{1}{4^s} + \cdots = \sum_{n=1}^{\infty} \frac{1}{n^s}. \tag{1.1}$$

By exploiting the convergent property of $\zeta(s)$, Euler proved that

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} = \prod_{p} \frac{1}{1 - p^{-s}}. \tag{1.2}$$

By this result he found a deep connection between $\zeta(s)$ and the distribution of primes. $\zeta(1)$ is the diverging harmonic series, yielding another proof that there are infinitely many prime numbers. Using the zeta function Euler was able to solve the so-called Basel problem which asks for the precise summation of the reciprocals of the squares of the natural numbers

$$\zeta(2) = \sum_{n=1}^{\infty} \frac{1}{n^2} = \lim_{n\to\infty} \left(\frac{1}{1^2} + \frac{1}{2^2} + \cdots + \frac{1}{n^2} \right) = \frac{\pi^2}{6}. \tag{1.3}$$

1.2 Distribution of primes

In the second half of the eighteenth century *Carl Friedrich Gauss* (1777-1855) used Euler's observation about the divergence of the harmonic sum of prime numbers and proved that

$$\sum_{p \le x} \frac{1}{p} \sim \log\log x\,, \tag{1.4}$$

and conjectured that the number of primes up to x is asymptotically

$$\pi(x) \sim \frac{x}{\log x}, \quad \text{as } x \to \infty \tag{1.5}$$

where $\pi(x)$ is the prime-counting function. Later Gauss refined his conjecture to

$$\pi(x) \sim Li(x) = \int_2^x \frac{1}{\log t} dt \tag{1.6}$$

where $Li(x)$ is the logarithmic integral. This connection between the primes and the natural logarithm (1.6) is known as the Prime Number Theorem (PNT) that describes the asymptotic distribution of the prime numbers among the positive integers. PNT was proved independently by Hadamard [4] and de la Vallée Poussin [5] in 1896 using the theory of the *Riemann zeta function*. In 1899 de la Vallée Poussin proved that the PNT is true in the following sharper form

$$\pi(x) = Li(x) + \mathcal{O}(xe^{-c\sqrt{\log x}}), \quad \text{as } x \to \infty, \quad \text{for some } c > 0. \tag{1.7}$$

The current best bound for the error term of the PNT is due to N.M Korobov [6] and I.M Vinogradov [7]. In 1958 they proved that for some positive c constant

$$\pi(x) = Li(x) + \mathcal{O}(xe^{-c\log x^{3/5}\log\log x^{-1/5}}), \quad \text{as } x \to \infty. \tag{1.8}$$

Aside from minor improvements, Vinogradov's and Korobov's result still represents the current record in the error term of the PNT.

1.2.1 Average gaps between primes

PNT states that the average gap between consecutive prime numbers among the first x integers is roughly $log(x)$.

Let p_n denote the nth prime number. In a breakthrough paper Dan Goldston, János Pintz, and Cem Yıldırım proved [8] in 2005 that for any positive number ϵ there exist primes p_{n+1} and p_n such that the difference between p_{n+1} and p_n is smaller than $\epsilon \log p_n$. In other words, there exist consecutive primes which are closer than any arbitrarily small multiple of the average spacing, that is,

$$\liminf_{n\to\infty} \frac{p_{n+1} - p_n}{\log p_n} = 0. \tag{1.9}$$

Based on the work of Goldston, Pintz, and Yildırım in 2013 Yitang Zhang found a proof [9] of the existence of infinitely many bounded gaps between primes. Formally,

$$\liminf_{n\to\infty}(p_{n+1} - p_n) < 7 \cdot 10^7. \tag{1.10}$$

James Maynard introduced a new method [10] to reduce the bound to 600. Using the techniques of Maynard, the Polymath 8 project improved the bound to 246. This is huge progress towards proving that there are infinitely many twin primes.

Considering the error term of the PNT one can claim a much stronger result assuming an interesting property of the distribution of the zeros of the *Riemann zeta function*.

1.3 The Riemann zeta function

Georg Friedrich Bernhard Riemann (1826-1866) recognized the importance of viewing $\zeta(s)$ as a function of a complex variable $s = \sigma + it$. This extended Euler zeta function is the Riemann zeta function which is an extremely important function of mathematics and physics. For all complex numbers s with $\sigma > 1$ the function can be defined by the integral

$$\zeta(s) = \sum_{n=1}^{\infty} n^{-s} = \frac{1}{\Gamma(s)} \int_0^{\infty} \frac{x^{s-1}}{e^x - 1} dx \tag{1.11}$$

where $\Gamma(s)$ is the gamma function

$$\Gamma(s) = \int_0^{\infty} e^{-x} x^{s-1} dx \quad s > 1 \tag{1.12}$$

and $\zeta(s)$ is a meromorphic function on the whole complex s-plane, which is holomorphic everywhere except for a simple pole at $s = 1$ with residue 1. By analytic continuation the function can be extended to the whole complex plane, except for $s = 1$, satisfying the functional equation

$$\zeta(s) = \chi(s)\zeta(1 - s) \tag{1.13}$$

where

$$\chi(s) = 2^s \pi^{s-1} \sin\left(\frac{s\pi}{2}\right) \Gamma(1 - s) \tag{1.14}$$

which can be written in a symmetric form

$$\Gamma\left(\frac{s}{2}\right) \pi^{-s/2} \zeta(s) = \Gamma\left(\frac{1 - s}{2}\right) \pi^{-(1-s)/2} \zeta(1 - s). \tag{1.15}$$

From the functional equation formula (1.14) $\sin\left(\frac{\pi s}{2}\right)$ shows that $\zeta(-2n) = 0\,(\forall n \in \mathbb{N})$. These are called trivial zeros of $\zeta(s)$. The negative even integers are not the only values for which the zeta function is zero. The other ones are called non-trivial zeros of $\zeta(s)$ and have a deep connection to the distribution of prime numbers.

The first few non-trivial zeros of $\zeta(s)$ accurate to within $4 * 10^{-9}$ are the following ones:

$\zeta(\frac{1}{2} + 14.134725142i)$, $\zeta(\frac{1}{2} + 21.022039639i)$, $\zeta(\frac{1}{2} + 25.010857580i)$, $\zeta(\frac{1}{2} + 30.424876126i)$.

One can observe that all the non-trivial zeros have $\Re(s) = 1/2$. This observation dates back to the nineteenth century.

1.3.1 The Riemann-hypothesis

In 1859 Bernhard Riemann conjectured [12] that all non-trivial zeros of $\zeta(s)$ have real part $\sigma = 1/2$. If the conjecture is correct, all the non-trivial zeros lie on the critical line $\frac{1}{2} + it$. This is the famous Riemann-hypothesis, one of the most important unsolved problems in the theory of prime numbers. The Riemann hypothesis is one of Hilbert's unsolved problems and is one of the Clay Mathematics Institute's Millennium Prize Problems.

$\zeta(s) = 0$, if $\Re(\zeta(s)) = 0$ and $\Im(\zeta(s)) = 0$ at the same time. The real part (black) and imaginary part (red) of the Riemann zeta function along the critical line $\Re(s) = 1/2$ can be seen in Figure (1.1).

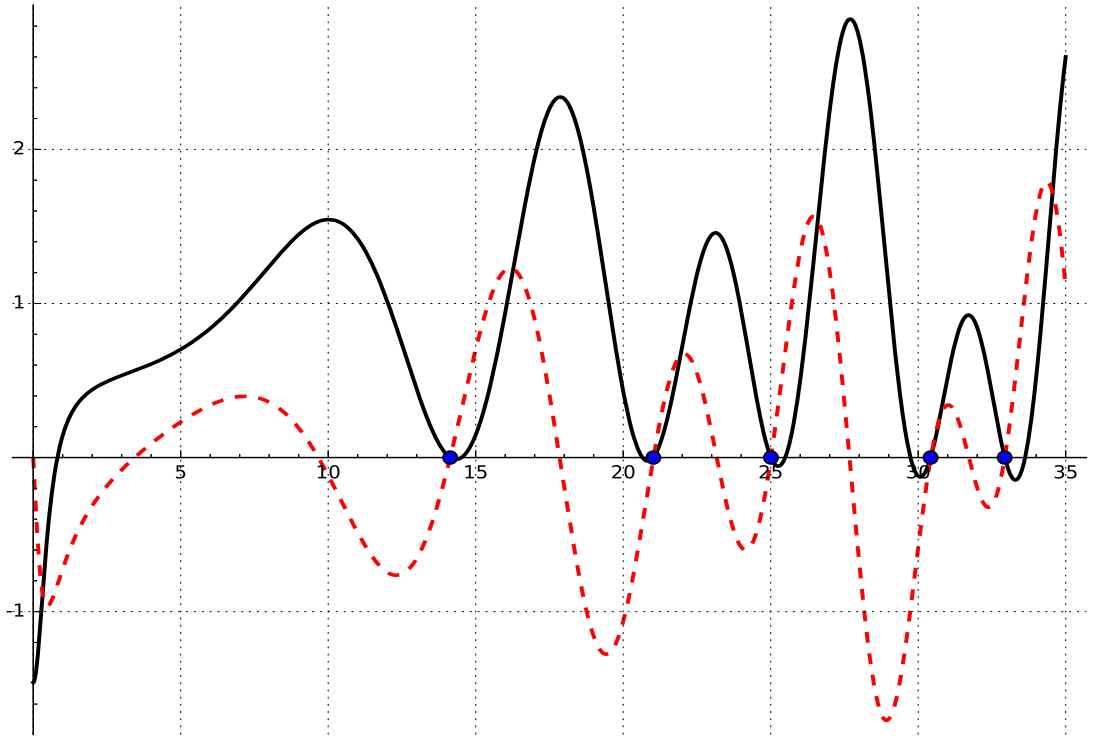

FIGURE 1.1: Non-trivial zeros of $\zeta(s)$ along the critical line $\Re(s) = 1/2$

We note that if ρ is a non-trivial zero of $\zeta(s)$, then so is $1-\rho$ by the functional equation. Furthermore, since $\overline{\zeta(s)} = \zeta(\overline{s})$, we deduce that $\overline{\rho}$ and $1 - \overline{\rho}$ are also zeros. We can conclude that the zeros are symmetrically arranged about the critical line given by $\sigma = 1/2$ and also about the real axis. For this reason we only calculate the zeros in the upper half plane.

Many authors computed the zeros of $\zeta(s)$ proving that the Riemann-hypothesis holds in some bounded region. In 1979 Richard P. Brent showed that the first 75 million

non-trivial zeros lie on the critical line [13]. In 1986 J. van de Lune, te Riele and Winter checked [14] that the first 1.5 billion non-trivial zeros all lie on the critical line. S. Wedeniwski used the distributed ZetaGrid project to verify that the first 100 billion zeros all lie on the critical line. In 2004 the Odlyzko-Schönhage algorithm [15] was used by Gourdon to verify the Riemann hypothesis for the first 10^{13} zeros of the zeta function [16].

1.3.2 Equivalent statements of the Riemann-hypothesis

Many statements equivalent to the Riemann hypothesis have been established and none of them have been proved or disproved. Without loss of generality we recall some famous statements that are equivalent to the RH and can be described in terms of the Riemann zeta function.

The Mertens function

The Möbius function $\mu(n)$ is an important multiplicative function in number theory. $\mu(n)$ is defined, for positive integers n, as

$$\mu(n) = \begin{cases} 1 & \text{if } n = 1, \\ (-1)^k & \text{if } n \text{ is a product of } k \text{ different prime numbers,} \\ 0 & \text{if } n \text{ has one or more repeated prime factors.} \end{cases}$$

The Mertens function is defined for all positive integers n by the summatory function of the Möbius function:

$$M(n) = \sum_{k=1}^{n} \mu(k). \tag{1.16}$$

In 1912 J. E. Littlewood [17] proved that the Riemann Hypothesis is equivalent to

$$|M(x)| = \mathcal{O}(x^{1/2+\varepsilon}), \quad \text{when } x \to \infty, \quad \text{for every } \varepsilon > 0\,. \tag{1.17}$$

Mertens conjectured that for $x > 1$

$$|M(x)| < \sqrt{x}. \tag{1.18}$$

This conjecture was disproved in 1985 by A. M. Odlyzko and H. te Riele [18] without providing an explicit counterexample. The disproof relies on extensive computations with the zeros of the zeta function. In 1987, J. Pintz [19] proved that the Mertens

conjecture is false for some $n < \exp(3.21 \times 10^{64})$. In 2006 T. Kotnik and H. te Riele [20] showed that the Mertens conjecture is false for some $n < \exp(1.59 \times 10^{40})$.

The Liouville function

Let $\omega(n)$ denote the number of prime factors of n, counted with multiplicity. Define Liouville's multiplicative function by the following formula:

$$\lambda(n) = (-1)^{\omega(n)}. \tag{1.19}$$

The Dirichlet series for the Liouville function is deeply related to the Riemann zeta function, namely

$$\sum_{n\geq 1} \frac{\lambda(n)}{n^s} = \frac{\zeta(2s)}{\zeta(s)}, \ \Re(s) > 1. \tag{1.20}$$

Based on numerical data, in 1919 Pólya conjectured [21] that below any bound N more than 50% of the natural numbers have an odd number of prime factors. In terms of the Liouville function the conjecture says that

$$L(n) = \sum_{k=1}^{n} \lambda(k) \leq 0 \tag{1.21}$$

holds for all $n > 1$. Pólya's conjecture would imply the Riemann hypothesis, however C. Brian Haselgrove in 1958 has shown that $L(n)$ changes sign infinitely often [22]. In 1960 Lehman found [23] that for $L(906\,180\,359) = +1$. The smallest counterexample is $n = 906\,150\,257$, found by Minoru Tanaka [24] in 1980. The Riemann hypothesis holds if and only if, for every $\epsilon > 0$,

$$\lim_{n\to\infty} \frac{\lambda(1) + \lambda(2) + \lambda(3) + \cdots + \lambda(n)}{n^{1/2+\epsilon}} = 0, \tag{1.22}$$

so a natural number n has equal probability of having an odd or even number of distinct prime factors (counted with multiplicity).

Error term of the PNT

The Riemann hypothesis implies much stronger result about the distribution of prime numbers than the prime number theorem. In 1901 Helge von Koch [25] proved that the Riemann hypothesis is equivalent to the error term in the prime number theorem having the bound

$$\pi(x) = Li(x) + \mathcal{O}(\sqrt{x}\log x). \tag{1.23}$$

More explicitly, the Riemann hypothesis implies

$$|\pi(x) - Li(x)| < \frac{1}{8\pi}\sqrt{x}\log x \tag{1.24}$$

for all $x \geq 2657$ [11].

1.3.3 Number of zeros of $\zeta(s)$

In the last century many partial results were achieved regarding the number of zeros of $\zeta(s)$.

Let $N(T)$ be the number of zeros of $\zeta(s)$ in $0 < t \leq T$ and let $N_0(T)$ be the number of such zeros with $\sigma = 1/2$.

The number of non-trivial zeros is asymptotically given by Riemann in his famous paper [12] from 1859,

$$N(T) = \frac{T}{2\pi}\log\left(\frac{T}{2\pi e}\right) + \frac{7}{8} + \mathcal{O}(\log T) \tag{1.25}$$

which was proved by von Mangoldt in 1895 [26]. The zeros of the zeta function become more and more dense as one goes upwards in the critical strip.

In 1914 G. H. Hardy proved that $\zeta(\frac{1}{2} + it)$ has infinitely many zeros [27], thus

$$N_0(T) \to \infty \quad \text{as } T \to \infty. \tag{1.26}$$

In 1921 Hardy and Littlewood showed [28] that there exist positive constants c and T_0, so that

$$N_0(T) > cT, \quad \text{for some } c > 0 \quad (T > T_0). \tag{1.27}$$

In 1942 Selberg proved that at least a small positive proportion of zeros lie on the critical line [29] and got a lower bound for $N_0(T)$

$$N_0(T) > cT\log(T), \quad \text{for some } c > 0 \quad (T > T_0). \tag{1.28}$$

In 1974 Levinson proved [30] that at least one third of the zeros of the Riemann zeta function are on the critical line and in 1989 J. B. Conrey proved that at least two fifths of the zeros of the Riemann zeta function are on the critical line [31].

1.4 Hardy Z-function

Locating zeros of $\zeta(s)$ on the critical line is essential to getting a better understanding of the distribution of prime numbers.

In order to avoid difficulties that are encountered by the pole of $\zeta(s)$ at $s = 1$, one can write

$$\xi(s) = \frac{1}{2}s(s-1)\pi^{-s/2}\Gamma(s/2)\zeta(s). \tag{1.29}$$

One can easily observe that the term $(s-1)$ in the ξ function definition eliminates the simple pole of $\zeta(s)$ at $s = 1$, thus ξ is an entire function. From the functional equation of $\zeta(s)$ we have $\xi(s) = \xi(1-s)$ which implies that $\xi(s)$ is real on the critical line and $\xi(s)$ has the same zeros as the zeta function on the critical strip.

From the definition of the ξ function, it is known that there are no non-trivial zeros outside the critical strip between the lines $\sigma = 0$ and $\sigma = 1$. One can prove the existence of zeros exactly on the real line between two points by checking numerically that $\xi(s)$ has opposite signs at these points.

One can consider a much easier function to investigate $\zeta(s)$ on the critical line. Hardy used (1.29) to introduce $Z(t)$. From (1.14) we have $\chi(s)\chi(1-s) = 1$ and by the reflection principle, it follows that $|\chi(\frac{1}{2} + it)| = 1$.

One can define the *Riemann–Siegel theta* function in terms of the χ function for real values of t by

$$\theta(t) = -\frac{1}{2}\arg\chi(\frac{1}{2} + it)\,, \tag{1.30}$$

so that

$$\chi(\frac{1}{2} + it) = e^{-2i\theta(t)}\,, \tag{1.31}$$

so one arrives at

$$\zeta\left(\frac{1}{2} + it\right) = Z(t)e^{-i\theta(t)}\,, \tag{1.32}$$

where $Z(t)$ is *Hardy's function* or the *Riemann-Siegel Z-function*. $\theta(t)$ can also be defined in terms of the Gamma function for real values of t by

$$\theta(t) = \arg\left(\Gamma\left(\frac{2it+1}{4}\right)\right) - \frac{\log\pi}{2}t \approx \frac{t}{2}\log\frac{t}{2\pi} - \frac{t}{2} - \frac{\pi}{8} + \frac{1}{48t} + \frac{7}{5760t^3} + \cdots \tag{1.33}$$

(1.33) is not convergent, but the first few terms give a good approximation for $t \gg 1$.

The connection between $\xi(s)$ and $Z(t)$ can be described (Ch. 4 pp. 89. in [32]) by

$$Z(t) = -2\pi^{\frac{1}{4}} \frac{\xi(\frac{1}{2}+it)}{(t^2+\frac{1}{4})|\Gamma(\frac{1}{4}+\frac{1}{2}it)|}. \tag{1.34}$$

Equation (1.32) implies that $Z(t)$ is real for real t and we have $|\zeta(1/2+it)| = |Z(t)|$. This behaviour of $Z(t)$ can be used to investigate the behaviour of $\zeta(s)$. Clearly, investigating a real-valued function is much easier than investigating $\zeta(s)$.

1.4.1 The Riemann-Siegel Formula

Inside the critical strip ($0 < \sigma < 1$) one can approximate $\zeta(s)$ by two sums.

Hardy and Littlewood proved that if $0 < \sigma < 1$, $s = \sigma + it$, $2\pi xy = t$, and $x, y > 0$ then

$$\zeta(s) = \sum_{n \le x} \frac{1}{n^s} + \chi(s) \sum_{n \le y} \frac{1}{n^{1-s}} + \mathcal{O}(x^{-\sigma}) + \mathcal{O}(t^{1/2-\sigma} y^{\sigma-1}). \tag{1.35}$$

This is known as the *approximate functional equation* (e.g: Ch. 4 pp. 79. in [32]). Focusing only one the critical line let $x = y = \{t/(2\pi)\}^{1/2}$.

Then (1.35) gives

$$\zeta(1/2+it) = \sum_{n \le x} n^{-1/2-it} + \chi(1/2+it) \sum_{n \le x} n^{-1/2+it} + \mathcal{O}(t^{-1/4}). \tag{1.36}$$

Multiplying (1.36) by $e^{i\theta(t)}$, we obtain

$$\zeta(1/2+it)e^{i\theta(t)} = e^{i\theta(t)} \sum_{n \le x} n^{-1/2-it} + e^{-i\theta(t)} \sum_{n \le x} n^{-1/2+it} + \mathcal{O}(t^{-1/4}), \tag{1.37}$$

thus,

$$Z(t) = 2 \sum_{n=1}^{\lfloor \sqrt{t/2\pi} \rfloor} n^{-1/2} \cos(\theta(t) - t \cdot \log n) + \mathcal{O}(t^{-1/4}). \tag{1.38}$$

This is the Riemann-Siegel Formula (see e.g:[13, 33]) which can be calculated in time complexity of $\mathcal{O}(t^{1/2})$.

Ghaith A. Hiary in 2011 presented a method [34] to compute $Z(t)$ in $\mathcal{O}(t^{2/5}), \mathcal{O}(t^{1/3})$ and $\mathcal{O}(t^{4/13})$ time complexities, respectively.

The $\mathcal{O}(t^{1/3})$ and $\mathcal{O}(t^{4/13})$ methods rely on the existence of efficient algorithms to compute quadratic exponential sums and cubic exponential sums with a relatively small cubic coefficient [34].

1.4.2 Gram's law

For $m \geq 0$ the m^{th} Gram point g_m can be defined by the unique solution of

$$\theta(g_m) = m\pi\,. \tag{1.39}$$

Gram's law is based on the observation that $Z(t)$ usually changes sign in the Gram intervals $G_j = [g_j, g_{j+1})$ for $j \geq 0$. A Gram point g_j is said to be "good" if $(-1)^j Z(g_j) > 0$ and "bad" otherwise [35]. A Gram block with length k is an interval $M_j = [g_j, g_{j+k})$ such that g_j and g_{j+k} are good Gram points and $g_{j+1}, \ldots g_{j+k-1}$ are bad Gram points for $k \geq 1$. The interval M_j satisfies Rosser's rule if $Z(t)$ has at least k zeros in M_j. Rosser's rule is violated infinitely often, but only for a small fraction of the Gram blocks. The first exception to the Rosser rule is at the $13\,999\,825^{th}$ Gram point.

1.5 Peak values of $Z(t)$

In 1979 Brent computed the first 75 000 000 zeros of $\zeta(s)$ and observed an unusually large $Z(t)$ (> 79.6) near the $70\,354\,406^{th}$ Gram point [13]. In all the cases, where an exception to Rosser's rule was observed, there was a large local maximum of $Z(t)$ nearby.

The Riemann-Siegel Z-function has also a very deep connection to the Riemann hypothesis. RH is equivalent to the statement that all local maxima of $Z(t)$ are positive and all local minima are negative, and it has been suggested that if a counterexample exists then it should be in the neighbourhood of unusually large peaks of $\zeta(1/2 + it)$. In other words if there exists a real number t_0 with $Z(t_0) \neq 0$ such that $Z(t)$ has either a positive local minimum or a negative local maximum at $t = t_0$, then RH is false [36].

Calculating peak values of $\zeta(s)$ on the critical line can reveal new interesting behaviour of the distribution of prime numbers.

1.5.1 Largest known values of $\zeta(s)$ on the critical line

With emerging computer technologies larger and larger values of $Z(t)$ can be calculated. In 1983 J. van de Lune found that for $t = 725177880629981.914$ we have $Z(t) \approx -453.9$. The first values where $|Z(t)| > 1000$ were found by Odlyzko [37] in 1989, the largest one at that time was the value $Z(t) \approx 1581.7$ for $t = 5032868769288289111.35$. Calculating $Z(t)$ is a very expensive task with the original Riemann-Siegel Formula, even using the Odlyzko-Schönhage algorithm [15]. Due to this fact $Z(t) \approx 1581.7$ was the largest known value for the next twenty years. In 2010, based on the searching method of Odlyzko and applying the $\mathcal{O}(t^{1/3})$ algorithm, Bober and Hiary were able to find many large values of $Z(t)$ [39]. They found that $Z(t)$ is approximately 16244.8652 for $t = 392467645898943091552511692841040506$. Finding peak values of $Z(t)$ is computationally expensive and challenging even with modern supercomputers. No explicit formula is known to find such t where $\zeta(1/2 + it)$ is large, however, methods are known from A.M. Odlyzko [37] and T. Kotnik [38] for locating large values of $Z(t)$ more efficiently than choosing t randomly.

1.5.2 The RS-PEAK algorithm in a nutshell

The main topic of this thesis is to present an efficient algorithm for finding extremely large $Z(t)$ candidates. The `RS-PEAK` algorithm [56] is based on simultaneous Diophantine approximations and can be used very effectively to find good candidates where large $Z(t)$ is likely. Prior to our publication only twelve $|Z(t)| > 10\,000$ values were found by Hiary and Bober. Applying the `RS-PEAK` algorithm we have found thousands of $|Z(t)| > 10\,000$ values and the largest known value

$$Z(3106788336290839656675405765936 82.05) \approx 16874.202$$

has been calculated recently. To the best of our knowledge, at the time of writing this thesis it is the largest $Z(t)$ ever calculated.

The `RS-PEAK` algorithm has three main parts:

- Part I - Fast Diophantine Approximation (will be presented in Chapter 2),
- Part II - Prefiltering (will be presented in Chapter 3),
- Part III - Main filtering (will be presented in Chapter 3).

The first part is for generating candidates where large $Z(t)$ is likely by solving simultaneous Diophantine approximations. The second and third parts are sieving methods

for eliminating weak candidates. In the next Chapter we are going to present the most important part of the `RS-PEAK` algorithm. We present how to solve n-dimensional Diophantine approximation problems in a very efficient way.

Chapter 2

Simultaneous Diophantine Approximation

Fast Diophantine approximation is one of the most important parts of the `RS-PEAK` algorithm. Recall the main summand of the Riemann-Siegel formula:

$$Z(t) = 2 \sum_{n=1}^{\lfloor\sqrt{t/2\pi}\rfloor} \frac{1}{\sqrt{n}} \cos(\theta(t) - t \cdot \log n) + \mathcal{O}(t^{-1/4}). \tag{2.1}$$

In this chapter we are focusing on the approximation of $\cos(\theta(t) - t \cdot \log n)$ for as many n as possible. In 1989 Andrew M. Odlyzko presented a method for predicting large values of $Z(t)$. "We need to find a t for which there exist integers $m_1, \ldots, m_n$ such that each of $t \log p_k - 2\pi m_k$ is small $(1 \leq k \leq n)$" [37].

In 1982 Arjen Lenstra, Hendrik Lenstra and László Lovász invented a polynomial time lattice basis reduction algorithm (LLL) that can be used for solving simultaneous Diophantine approximations [40]. The main result of this chapter is to present `MAFRA` - *Multithreaded Advanced Fast Rational Approximation* algorithm. Applying `MAFRA` one can achieve significant improvement on the approximation of $\cos(\theta(t) - t \cdot \log n)$ which is much faster than LLL for small dimensions ($n < 20$). In our particular case the algorithm is used for efficiently generating candidates where large $Z(t)$ is expected.

This chapter is based on the following research papers: [57] , [58] , [59].

2.1 Historical overview

Rational approximation, or alternatively, Diophantine approximation is very important in many fields of mathematics and computer science. While the work of John Wallis

(1616–1703) and Christiaan Huygens (1629–1695) established the field of continued fractions, it began to blossom when Leonhard Euler (1707–1783), Johann Heinrich Lambert (1728–1777) and Joseph Louis Lagrange (1736–1813) embraced the topic. In the 1840s, Joseph Liouville (1809–1882) obtained an important result on general algebraic numbers. If α is an irrational algebraic number of degree n over the rational numbers, then there exists a constant $c(\alpha) > 0$ such that

$$\left|\alpha - \frac{p}{q}\right| > \frac{c(\alpha)}{q^n} \tag{2.2}$$

holds for all integers p and $q > 0$. This result allowed him to produce the first proven examples of transcendental numbers. In 1891 Adolf Hurwitz (1859–1919) proved that for each irrational α infinitely many pairs (p, q) of integers satisfy

$$\left|\alpha - \frac{p}{q}\right| < \frac{1}{q^2\sqrt{5}} \tag{2.3}$$

but there are some irrational numbers β for which at most finitely many pairs satisfy

$$\left|\beta - \frac{p}{q}\right| < \frac{1}{q^{2+\gamma}\sqrt{5+\mu}} \tag{2.4}$$

no matter how small the positive increments γ and μ are. Suppose α is a real algebraic number of degree $d \geq 2$. Liouville's theorem (2.2) implies that the inequality

$$\left|\alpha - \frac{p}{q}\right| < \frac{1}{q^{\mu}} \tag{2.5}$$

has only finitely many rational solutions p/q if $\mu > d$. In 1909 Thue showed that (2.5) has only finitely many solutions if $\mu > \frac{1}{2}d + 1$. In 1955 Roth proved that (2.5) has only finitely many solutions if $\mu > 2$.

The idea can be generalized to simultaneous approximation. Simultaneous Diophantine approximation originally means that for given real numbers $\alpha_1, \alpha_2, \ldots, \alpha_n$ find $p_1, p_2, \ldots, p_n, q \in \mathbb{Z}$ such that

$$\left|\alpha_i - \frac{p_i}{q}\right| \tag{2.6}$$

is "small" for all i, and q is "not too large".

For a given real α let us denote the nearest integer distance function by $\|.\|$, that is,

$$\|\alpha\| = \min\{|\alpha - j| : j \in \mathbb{Z}\} \tag{2.7}$$

Then, simultaneous approximation can be interpreted as minimizing

$$\max \{\|q\alpha_1\|, \ldots, \|q\alpha_n\|\}. \tag{2.8}$$

In 1842 Peter Gustav L. Dirichlet (1805–1859) showed that there exist simultaneous Diophantine approximations with absolute error bound $q^{-(1+1/n)}$. To be more precise, he showed that there are infinitely many approximations satisfying

$$|q \cdot \alpha_i - p_i| < \frac{1}{q^{1/n}} \tag{2.9}$$

for all $1 \leq i \leq n$. Unfortunately, no polynomial algorithm is known for the simultaneous Diophantine approximation problem. However, due to the LLL algorithm of Lenstra, Lenstra and Lovász, if $\alpha_1, \alpha_2, \ldots, \alpha_n$ are irrationals and $0 < \varepsilon < 1$ then there is a polynomial time algorithm to compute integers $p_1, p_2, \ldots, p_n, q \in \mathbb{Z}$ such that

$$1 \leq q \leq 2^{n(n+1)/4}\varepsilon^{-n} \text{ and } |q \cdot \alpha_i - p_i| < \varepsilon \tag{2.10}$$

for all $1 \leq i \leq n$ [40].

Lagarias [47, 48] presented many results concerning the best simultaneous approximations. Szekeres and T. Sós [50] analyzed the signatures of the best approximation vectors. Kim et al. [45] discussed rational approximations to pairs of irrational numbers which are linearly independent over the rationals and applications to the theory of dynamical systems. Armknecht et al. [41] used the inhomogeneous simultaneous approximation problem for designing cryptographic schemes. Lagarias [49] discussed the computational complexity of Diophantine approximation problems, which, depending on the specification, varies from polynomial-time to $\mathcal{NP}$-complete. Frank and Tardos [42] developed a general method in combinatorial optimization using simultaneous Diophantine approximations which could transform some polynomial time algorithms into strongly polynomial.

2.2 Problem statement

Consider the set of irrationals $\Upsilon = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$. Let $\varepsilon > 0$ and $1 \leq a \leq b$ be natural numbers. Furthermore, let us define the set

$$\Omega = \Omega(\Upsilon, \varepsilon, a, b) = \{k \in \mathbb{N} : a \leq k \leq b, \|k\alpha_i\| < \varepsilon \text{ for all } \alpha_i \in \Upsilon\}. \tag{2.11}$$

For given Υ, ε and a, b the following computational challenges can be stated:

1. Determine all the elements of $\Omega(\Upsilon, \varepsilon, a, b)$,
2. Determine as many elements of $\Omega(\Upsilon, \varepsilon, a, b)$ as possible in a given time frame.

We refer to the first problem as the "all-elements simultaneous Diophantine approximation problem". In case of $|\Upsilon| = n \geq 1$ we call it an n-dimensional simultaneous approximation. The second problem is referred to as the "approximating as many elements as possible" problem. Let us define the following two exact challenges:

2.2.1 1-dimensional challenge

Determine all elements of

$$\Omega\big(\{\sqrt{2}\}, 10^{-17}, 10^{20}, 10^{21}\big). \tag{2.12}$$

2.2.2 n-dimensional challenge

Determine as many elements of Ω as possible in a given time frame.

$$\Omega\left(\left\{\frac{\log(p)}{\log(2)}, p \text{ prime}, 3 \leq p \leq 19\right\}, 10^{-2}, 1, 10^{18}\right). \tag{2.13}$$

Note: The above mentioned computational challenges are deeply connected to the generation of t candidates where large $Z(t)$ is expected. Recall the main summand of the Riemann-Siegel formula:

$$Z(t) = 2 \sum_{n=1}^{\lfloor\sqrt{t/2\pi}\rfloor} \frac{1}{\sqrt{n}} \cos(\theta(t) - t \cdot \log n) + \mathcal{O}(t^{-1/4}). \tag{2.14}$$

For $t = \frac{2k\pi}{\log 2}$ we have

$$\cos\left(\theta(t) - \frac{2k\pi}{\log 2} \cdot \log n\right). \tag{2.15}$$

Approximating (2.15) well with $\cos(\theta(t))$ for as many n as possible is a simultaneous Diophantine approximation problem (further and detailed explanation of the connection between (2.13) and (2.15) will be presented in Chapter 3).

Note that the approximation of (2.13) and (2.15) are equivalent approximation problems. In this chapter we are focusing on the efficient approximation of (2.13). If one can solve (2.13) efficiently, then one can solve (2.15) efficiently either. Methods presented in this chapter will be used in Chapter 3 to generate candidates where large $Z(t)$ is expected.

2.3 The continued fraction approach

It is well-known that continued fractions are one of the most effective tools of rational approximation to a real number. *Simple continued fractions* are expressions of the form

$$a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cdots}} \tag{2.16}$$

where a_i are integer numbers with $a_1, a_2, \ldots > 0$. It is called *finite* if it terminates, and *infinite* otherwise. These continued fractions are usually represented in bracket form $[a_0, a_1, \ldots, a_m, \ldots]$, i.e.

$$C_0 = [a_0] = a_0, \quad C_1 = [a_0, a_1] = a_0 + \frac{1}{a_1}, \quad C_2 = [a_0, a_1, a_2] = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2}}, \quad \ldots \tag{2.17}$$

where the C_m are called *convergents*. Clearly, the convergents C_m represent some rational numbers p_m/q_m. An infinite continued fraction $[a_0, a_1, a_2, \ldots]$ is called convergent if its sequence of convergents C_m converges in the usual sense, i.e. the limit

$$\alpha = \lim_{m\to\infty} C_m = \lim_{m\to\infty} [a_0, a_1, \ldots, a_m] \tag{2.18}$$

exists. In this case we say that the continued fraction represents the real number α. The simple continued fraction expansion of $\alpha \in \mathbb{R}$ is infinite if and only if α is irrational. The convergents C_m are the best rational approximations in the following sense:

Lemma 2.1. *No better rational approximation exists to the irrational number α with smaller denominator than the convergents $C_m = p_m/q_m$ (e.g: Ch. IV.12 in [43]) .*

The simple continued fraction approximation for $\sqrt{2}$ is $[1, 2, 2, \ldots]$, the sequence of the convergents is

$$1, \frac{3}{2}, \frac{7}{5}, \frac{17}{12}, \frac{41}{29}, \frac{99}{70}, \frac{239}{169}, \frac{577}{408}, \frac{1393}{985}, \frac{3363}{2378}, \frac{8119}{5741}, \ldots \tag{2.19}$$

Among all fractions with denominator at most 29, the fraction $41/29$ is the closest to $\sqrt{2}$, among all fractions with denominator at most 70, the fraction $99/70$ is closest to $\sqrt{2}$, and so on.

Every convergent is a best rational approximation, but these are not all of the best rational approximations. Fractions of the form

$$\frac{p_{m-1} + jp_m}{q_{m-1} + jq_m} \quad (1 \le j \le a_{m+2} - 1) \tag{2.20}$$

are called *intermediate convergents* or *semi-convergents*. To get every rational approximation between two consecutive p_m/q_m and p_{m+1}/q_{m+1}, we have to calculate the intermediate convergents.

The missing intermediate convergents in (2.19) are

$$\frac{4}{3}, \frac{10}{7}, \frac{24}{17}, \frac{58}{41}, \frac{140}{99}, \frac{338}{239}, \frac{816}{577}, \frac{1970}{1393}, \frac{4756}{3363}, \dots \tag{2.21}$$

The approximations $|\alpha - p/q|$ above are also known as "best rational approximations of the first kind". However, sometimes we are interested in the approximations $|\alpha \cdot q - p|$. This is called the *approximation of a second kind*.

Lemma 2.2. [44] *A rational number p/q, which is not an integer, is a convergent of a real number α if and only if it is a best approximation of the second kind of α.*

In 1997 Clark Kimberling proved [46] the following result regarding intermediate convergents:

Theorem 2.3. *The best lower (upper) approximates to a positive irrational number α are the even-indexed (odd-indexed) intermediate convergents.*

In order to generate many integers q that satisfy

$$\|q \cdot \sqrt{2}\| < 10^{-5} \tag{2.22}$$

one can apply the theory of continued fractions, especially convergents. If q_m is the first integer that satisfies $\|q_m \cdot \sqrt{2}\| < 10^{-5}$ in the continued fraction expansion of $\sqrt{2}$, then all convergents with denominator larger than q_m will satisfy equation (2.22).

Consider the 1-dimensional challenge stated in (2.12). There are only 3 convergents of $\sqrt{2}$ where $10^{20} < q_m < 10^{21}$. They are

$$\frac{233806732499933208099}{165326326037771920630}, \frac{564459384575477049359}{399133058537705128729}, \frac{1362725501650887306817}{963592443113182178088}.$$

With intermediate convergents we get 2 more solutions. Hence, with the theory of continued fractions we are able to find only 5 appropriate integers. One may ask how many elements are in the set Ω in (2.12)?

Hermann Weyl (1855–1955) and Waclaw Sierpiński (1882–1969) proved in 1910 that if $\alpha \in \mathbb{R} \setminus \mathbb{Q}$ then $\alpha, 2\alpha, 3\alpha, \dots \pmod 1$ is uniformly distributed on the unit interval. From

this theorem it immediately follows that there are approximately $2(b-a)\varepsilon$ appropriate integers in the $[a,b]$ interval. In Challenge (2.12) we expect $2(10^{21}-10^{20})\cdot 10^{-17} = 18000 \pm 1$ integers. This is by several orders of magnitude more than what we were able to obtain by continued fractions.

2.3.1 The Lenstra–Lenstra–Lovász approach

We have seen in the previous section that Challenge (2.12) is unsolvable with the theory of continued fractions. Challenge (2.13) is a 7-dimensional simultaneous approximation problem and is even more beyond the potential of continued fractions. Although there is no known polynomial-time algorithm that is able to solve the Dirichlet type simultaneous Diophantine approximation problem, there exists an algorithm that can be useful for similar problems. The Lenstra–Lenstra–Lovász basis reduction algorithm (*LLL*) is a polynomial-time algorithm that finds a reduced basis in a lattice [40]. The algorithm can be applied to solve simultaneous Diophantine approximation problems *with an extra condition.*

Theorem 2.4. *There exists a polynomial-time algorithm for the given irrationals* $\alpha_1, \alpha_2, \ldots, \alpha_n$ *and* $0 < \varepsilon < 1$ *that can compute integers* $p_1, \ldots, p_n$ *and* q *such that*

$$\left|\alpha_i - \frac{p_i}{q}\right| < \frac{\varepsilon}{q}$$

and

$$0 < q \leq \beta^{n(n+1)/4}\varepsilon^{-n}$$

hold for all $1 \leq i \leq n$*, where* β *is an appropriate reduction parameter.*

The extra condition is the bound $0 < q \leq \beta^{n(n+1)/4}\varepsilon^{-n}$.

In one dimension the *LLL* algorithm provides exactly the continued fraction approach discussed in the previous section. To find all 18000 solutions for Challenge (2.12) we have to consider other possibilities. What about Challenge (2.13)?

Let $\alpha_1, \alpha_2, \ldots, \alpha_n$ be irrational numbers and let us approximate them with rationals admitting an $\varepsilon > 0$ error. Let $X = \beta^{n(n+1)/4}\varepsilon^{-n}$ and let the matrix A be the following:

$$A = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ \alpha_1 X & X & 0 & \ldots & 0 \\ \alpha_2 X & 0 & X & \ldots & 0 \\ \vdots & & & & \vdots \\ \\ \alpha_n X & 0 & 0 & \ldots & X \end{bmatrix}.$$

Applying the LLL algorithm for A, the first column of the resulting matrix contains the vector $[q, p_1, p_2, p_3, \ldots, p_n]^T$ which satisfies Theorem (2.4).

Let us see how the LLL algorithm works in dimension 7 for solving Challenge (2.13). Let $\alpha_i = \frac{\log p_{i+1}}{\log(2)}$ where p_i denotes the i-th prime for $1 \leq i \leq 7$, and let $\varepsilon = 0.01$. We are looking for an integer $q \leq 2^{14} \cdot 100^7$ that satisfies $\|q \cdot \alpha_i\| < \varepsilon$ for all i. Applying the LLL algorithm we get $q = 1325886000944418$. It is easy to verify that $\|q\,\alpha_i\| < 0.01$ holds for all $1 \leq i \leq 7$.

The real drawback of the method for our purposes is that it is inappropriate for finding *all* or *many* different solutions q in an arbitrary interval.

We note that sometimes one can find a few more solutions with a different choice of β.

It can be concluded that the apparatus of the continued fractions is not appropriate for solving Challenge (2.12) and Challenge (2.13) type problems. The LLL can be used to solve approximation problems like (2.12) or (2.13), however we are focusing on finding all possible solutions as fast as possible.

In the following section we are going to provide an effective solution for Challenge (2.12) and Challenge (2.13) that overperforms LLL.

2.4 Approximations in the one-dimensional case

2.4.1 "All-elements" approximation

In this section we present how to calculate all the elements of $\Omega(\Upsilon, \varepsilon, a, b)$ where $\Upsilon = \{\alpha\}$.

For a given Ω let $k : \{1, 2, \ldots, |\Omega|\} \to \Omega$ monotonically increasing, so k_i denotes the ith integer in Ω. Let us define the set

$$\Delta_\Omega = \{k_{n+1} - k_n : \ 1 \leq n \leq |\Omega| - 1\}. \tag{2.23}$$

The set Δ_Ω contains all possible step-sizes between two consecutive k_i's. We have the following important result regarding the cardinality of Δ_Ω.

Theorem 2.5 ([57])**.** $|\Delta_\Omega| \leq 3$.

Proof. The proof has two parts. In the first step we construct all the possible three elements of Δ_Ω and in the second step we show that there is no more. For the given

irrational α and an arbitrary $m \in \mathbb{N}$ let

$$\langle m \rangle = \begin{cases} \|\alpha m\| & \text{if } \alpha m - \|\alpha m\| \in \mathbb{N}\,, \\ -\|\alpha m\| & \text{if } \alpha m + \|\alpha m\| \in \mathbb{N}\,. \end{cases} \tag{2.24}$$

Let us furthermore define the following open intervals:

$$A = (-2\varepsilon, -\varepsilon),\ B = (-\varepsilon, 0),\ C = (0, \varepsilon),\ D = (\varepsilon, 2\varepsilon)\,. \tag{2.25}$$

Let m_1 be the smallest positive integer that satisfies $\langle m_1 \rangle \in C \cup D$, let m_2 be the the smallest positive integer that satisfies $\langle m_2 \rangle \in A \cup B$ and let $m_3 = m_1 + m_2$.

The first part of the proof is to show that there is always at least one integer (m_1, m_2 or m_3) which adding to an arbitrary $k_i \in \Omega$ always produces a new integer $k_j \in \Omega$. Clearly, $\langle k_i \rangle \in B \cup C$ for all k_i. Let us see the following cases:

$\langle k_i \rangle \in B$:

If $\langle m_1 \rangle \in C$, $\langle m_2 \rangle \in A \cup B$ then $\langle k_i + m_1 \rangle \in B \cup C$.

If $\langle m_1 \rangle \in D$, $\langle m_2 \rangle \in A$ and $\langle m_1 + m_2 \rangle \in C$ then $\langle k_i + (m_1 + m_2) \rangle \in B \cup C$.

If $\langle m_1 \rangle \in D$, $\langle m_2 \rangle \in A$ and $\langle m_1 + m_2 \rangle \in B$ then $\langle k_i + (m_1 + m_2) \rangle \in A \cup B$.

If $\langle k_i + (m_1 + m_2) \rangle \in A$ then $\langle k_i + (m_1 + m_2) - m_2 \rangle \in B \cup C$.

If $\langle m_1 \rangle \in D$, $\langle m_2 \rangle \in B$ and $\langle m_1 + m_2 \rangle \in C$ then $\langle k_i + (m_1 + m_2) \rangle \in B \cup C$.

If $\langle m_1 \rangle \in D$, $\langle m_2 \rangle \in B$ and $\langle m_1 + m_2 \rangle \in D$ then $\langle k_i + (m_1 + m_2) \rangle \in C \cup D$.

If $\langle k_i + (m_1 + m_2) \rangle \in D$ then $\langle k_i + (m_1 + m_2) - m_1 \rangle \in B \cup C$.

$\langle k_i \rangle \in C$:

If $\langle m_1 \rangle \in C \cup D$, $\langle m_2 \rangle \in B$ then $\langle k_i + m_2 \rangle \in B \cup C$.

If $\langle m_1 \rangle \in C$, $\langle m_2 \rangle \in A$ and $\langle m_1 + m_2 \rangle \in B$ then $\langle k_i + (m_1 + m_2) \rangle \in B \cup C$.

If $\langle m_1 \rangle \in C$, $\langle m_2 \rangle \in A$ and $\langle m_1 + m_2 \rangle \in A$ then $\langle k_i + (m_1 + m_2) \rangle \in A \cup B$.

If $\langle k_i + (m_1 + m_2) \rangle \in A$ then $\langle k_i + (m_1 + m_2) - m_2 \rangle \in B \cup C$.

If $\langle m_1 \rangle \in D$, $\langle m_2 \rangle \in A$ and $\langle m_1 + m_2 \rangle \in B$ then $\langle k_i + (m_1 + m_2) \rangle \in B \cup C$.

If $\langle m_1 \rangle \in D$, $\langle m_2 \rangle \in A$ and $\langle m_1 + m_2 \rangle \in C$ then $\langle k_i + (m_1 + m_2) \rangle \in C \cup D$.

If $\langle k_i + (m_1 + m_2) \rangle \in D$ then $\langle k_i + (m_1 + m_2) - m_1 \rangle \in B \cup C$.

Let now $X = \Delta_\Omega \setminus \{m_1, m_2, m_3\}$. We claim that $X = \varnothing$. Suppose otherwise, and let j be the smallest index with $m = k_{j+1} - k_j \in X$. Clearly, $\langle m \rangle \in A \cup B \cup C \cup D$. We can observe as well that for all $m \in \mathbb{N}$, $k_i \in \Omega$, $\langle k_i + m \rangle \in B \cup C$ implies $\langle m \rangle \in A \cup B \cup C \cup D$. Then it is easy to see that

- $j > 1$, and k_i's are integer linear combinations of m_1 and m_2 for all $i \leq j$,
- $m_1, m_2 < m < m_1 + m_2$,
- $\langle m \rangle \in A \cup D$.

If $\langle m \rangle \in A$ then $\langle m - m_2 \rangle \in B \cup C$, which contradicts the minimality of j. In the same way, if $\langle m \rangle \in D$ then $\langle m - m_1 \rangle \in B \cup C$, which is a contradiction again. Hence, such an m does not exist. The proof is complete. □

Note: We note that Theorem 2.5 is very deeply connected to the Steinhaus conjecture also known as the so-called "three-gap theorem" which was proven by Vera T. Sós [51], S. Świerckowski [52] and J. Surányi [53] .

Finding the integers m_1, m_2 and m_3 can be done very efficiently with the theory of intermediate convergents. We know from Theorem 2.3 that intermediate convergents of an irrational α always produce the best upper and lower approximations to α, so m_1 and m_2 must be intermediate convergents. Based on this result we created the algorithm `FindMMM` [57] .

Applying the `FindMMM` algorithm (see Algorithm A.1 in Appendix) for Challenge (2.12) we get the following values

$$\begin{aligned} m_1 &= 59341817924539925, \\ m_2 &= 24580185800219268, \\ m_3 &= 83922003724759193. \end{aligned}$$

After the precalculation of m_1 and m_2 it is very easy to compute every k_i between 10^{20} and 10^{21}. First we have to find an intermediate convergent between 10^{20} and 10^{21}. It is easy to do with the theory of continued fractions (e.g: 233806732499933208099). After that we can add, subtract m_1, m_2 or m_3 until we reach the bounds of the interval. The Weyl equidistribution theorem predicts 18000 integers that solve Challenge (2.12). Applying `Challenge 1 Solver` algorithm (see Algorithm A.2 in Appendix) we found exactly 18 000 integers. The precalculation and the computation of all k_i values took only 31 ms on a single Intel® Core i5-2450M Desktop PC .

2.4.2 "Many elements" approximation

In some cases it is not necessary to find all the k_i elements of Ω, rather it is enough to find as much as possible within a given time frame. Then, the following procedure works:

Find the smallest integer x that satisfies $0 < \langle x \rangle < \varepsilon$ and find the smallest integer y that satisfies $-\varepsilon < \langle y \rangle < 0$. Using the notations (2.25) it is easy to see that if $\langle k_i \rangle \in B$ and $\langle x \rangle \in C$ then $\langle k_i + x \rangle \in B \cup C$. In the same way, if $\langle k_i \rangle \in C$ and $\langle y \rangle \in B$ then $\langle k_i + y \rangle \in B \cup C$. Only with these two integers it is always possible to produce a subset of Ω.

If we want to determine just "many" elements of Ω, the previous method generates 12945 integers within 15 ms on a single Intel® Core i5-2450M Desktop PC

2.5 Approximations for the multi-dimensional case

2.5.1 "Many elements" approximation

Calculating all-elements of Ω seems to be hard in higher dimensions. However, we can generalize our one dimensional method to find "many" $q \in \Omega$ integers recursively. The method is based on the following Theorem:

Theorem 2.6 ([57]). *Let $\Upsilon = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ be a set of irrationals and $\varepsilon > 0$. Then there is a set $|\Gamma_n| = 2^n$ with the following property: if $q \in \Omega$ then $(q + \gamma) \in \Omega$ for some $\gamma \in \Gamma_n$.*

Proof. Let $q \in \Omega$ be given. Let us define an n-dimensional binary vector b associated with q in the following way:

$$b_i = \begin{cases} 1 & \text{if } q\alpha_i - \|q\alpha_i\| \in \mathbb{N}\,, \\ 0 & \text{if } q\alpha_i + \|q\alpha_i\| \in \mathbb{N}\ . \end{cases} \tag{2.26}$$

Let Γ_n be the set for which

1. $\gamma \in \Gamma_n$ implies $\|q\alpha_i\| < \varepsilon$ for all $1 \leq i \leq n$,
2. All the associated binary representations of (2.26) are different.

Then, for a given $q \in \Omega$ there exists a $\gamma \in \Gamma_n$ such that $q + \gamma \in \Omega$, e.g., when their associated binary representations are (1's binary) complements. Clearly, $|\Gamma_n| = 2^n$. The proof is finished. □

Remark 2.7. Remember the first dimension case: For all $m \in \mathbb{N}$, $q \in \Omega$, $\langle q+m \rangle \in B \cup C$ implies that $\langle m \rangle \in A \cup B \cup C \cup D$. We can generalize this to higher dimensions. Let $q \in \Omega$ and $m \in \mathbb{N}$ be given. Then $q + m \in \Omega$ implies $\|m \cdot \alpha_i\| \in A \cup B \cup C \cup D$ for all $1 \leq i \leq n$.

We have created an algorithm called `Precalc` [57] (see Algorithm A.5 in Appendix) that can be used efficiently for the precomputation of Γ_n, a subset of Δ_Ω. Unfortunately, the precalculation of 2^n integers is computationally expensive.

To make the generation even faster we have created the `Reduce` [57] algorithm (see Algorithm A.4 in Appendix A.). The algorithm uses sorting and indexing techniques to reduce the computation time of Γ_n.

In Challenge (2.13) the precalculation of the $2^7 = 128$ integers took approximately 6.14 seconds on a single Intel® Core i5-2450M Desktop PC.

Applying the `Challenge 2 Solver` algorithm (see Algorithm A.3 in Appendix) we were able to produce 120 852 integers in Ω within 26.8 sec.

2.6 Multidimensional case without boundaries

Let us define a new set like Ω (2.11) but without boundaries,

$$\Lambda(\Upsilon, \varepsilon) = \{k \in \mathbb{N} : \|k\alpha_i\| < \varepsilon \text{ for all } \alpha_i \in \Upsilon\}. \tag{2.27}$$

This "small" change of the definition allows us to design and develop an even faster algorithm. Checking the boundaries in every round of the loop is a very expensive task. We have modified the `Challenge 2 Solver` algorithm and created a general purpose rational approximation algorithm called `FRA - Fast Rational Approximation` (see Algorithm A.6 in Appendix).

Consider the following 10-dimensional simultaneous Diophantine approximation problem:

$$\Lambda\left(\left\{\frac{\log(p)}{\log(2)}, p \text{ prime}, 3 \leq p \leq 31\right\}, 0.01\right) \tag{2.28}$$

Determine 1 billion elements of the set as fast as possible.

Applying the `FRA` algorithm, Challenge (2.28) can be solved very efficiently using a sophisticated and well-optimized native C source code with the GNU MP 5.1.3 multi precision library.

It was possible to produce 100 000 integers within 2.65 seconds on a single Intel® Core i5-2450M Desktop PC.

2.6.1 AFRA – Advanced Fast Rational Approximation

Let $k \in \Lambda$. `FRA` always finds the smallest $\gamma \in \Gamma_{10}$ where $(k+\gamma) \in \Lambda$. It is easy to see that in the worst case this algorithm goes through all the 1024 elements of Γ_{10} (see Algorithm A.6, line 5). In each step the algorithm has to check whether $(k+\gamma) \in \Lambda$ or not (see Algorithm A.6, line 9).

We have created `AFRA - Advanced Fast Rational Approximation`. It finds one element from Γ_{10} — not necessary the smallest one[1] — that satisfies $(k+\gamma) \in \Lambda$ (Theorem 2.6 ensures finding the appropriate $\gamma \in \Gamma_{10}$ efficiently). `AFRA` is therefore faster, however, adding some γ to k produces larger values in Λ.

It can be concluded that `FRA` is a better choice for solving bounded challenges like $\Omega(\Upsilon, \varepsilon, a, b)$. For solving unbounded challenges, like $\Lambda(\Upsilon, \varepsilon)$, `AFRA` is the appropriate choice. Applying `AFRA`, it was possible to produce 100 000 integers $\in \Lambda$ within 0.434 seconds on a single Intel® Core i5-2450M Desktop PC.[2] This is almost ten times faster than the Algorithm 1 implementation.

Let us compare the algorithms `FRA` and `AFRA` with exact numbers. Consider the following challenge: generate as many integers as possible in the set

$$\Omega\left(\left\{\frac{\log(p)}{\log(2)}, p \text{ prime}, 3 \leq p \leq 31\right\}, 0.01, 0, 2 \times 10^{19}\right). \tag{2.29}$$

As we mentioned `FRA` is a better choice for a bounded challenge. Solving (2.29) by `FRA` one can produce 13 different integers between 0 and 2×10^{19}. These integers are presented in Table 2.1. It is easy to verify that every integer k in Table 2.1 satisfies the following:

$$\left\| k\frac{\log(p)}{\log(2)} \right\| < 0.01 \tag{2.30}$$

for all $3 \leq p \leq 31$.

These integers were generated in 0.015 seconds. `AFRA` is almost 10 times faster than `FRA`, however, inappropriate for solving this particular "bounded" challenge. With `AFRA` we

[1] The set of integers in Γ_{10} are ordered in the following way: every integer in Γ_n is represented by an n-dimensional binary vector. Γ_{10} contains integers ordered by the values of this binary vector (e.g: 0000000000, 0000000001, 0000000010, 0000000011 etc.)

[2] During the measurements Input/Output costs are not cumulated. Displaying the 100 000 integers from the memory would take approximately 5-6 seconds.

TABLE 2.1: FRA output between 0 and 2×10^{19}

102331725988392788	479125648045771184	710080108123034500
1711993379226146170	2088787301283524566	3423106890630466630
5441342799508541730	7540063840126351339	8406797017385611672
10118790396611757842	10503998465875331568	11021951848184774212
19036050657750584878		

can produce only one integer solution, which is 2298677471355273619. The next integer would be 183963121486836331196 which is already out of the upper bound (2×10^{19}).

We conclude that when the size of the integers is unimportant then Algorithm `AFRA` is the right solution.

2.6.2 Multithreaded AFRA

To make the generation even faster we modified our C code in order to be able to run tasks in parallel using pthreads (IEEE Std. 1003.1c-1995.). We refer to the multithreaded version of `AFRA` as `MAFRA - Multithreaded Advanced Fast Rational Approximation` algorithm.

In this section we present the measured running time of `MAFRA` for different architectures. The first test environment was a simple Sandy Bridge Intel® Core i5-2450M with 4 GB RAM. The second hardware was a Super Computing Cluster called ATLAS with 90x Intel® Xeon® E5520 Nehalem Quad Core 2.26 GHz Processors. The third hardware was an ATI Radeon 7970 GPU card.

2.6.2.1 Test – Core i5-2450M Laptop

Our first test environment was a single desktop PC. It was an Intel® Core i5-2450M Sandy Bridge CPU with 4 GB RAM having 2 cores. Generating 100 000 integers $\in \Lambda$ for solving the 10 dimensional challenge with the algorithm `MAFRA` took 0.234 sec. Our newly implemented, optimized and multithreaded C code is effective, however, generating 1 billion elements of (2.29) with this architecture would take approximately 39 minutes.

2.6.2.2 Test – ATLAS Computing Cluster

Our second test environment was the ATLAS Supercomputing Cluster. ATLAS is a high performance computing cluster operating at Eötvös Loránd University, Hungary.

ATLAS architecture consists of 11 HP ProLiant SL6000 Scalable systems with 22 HP SL2x170Z G6 server trays. The most important characteristics of ATLAS are the following:

Headnode:

1. 2x Intel® Xeon® E5520 Nehalem Quad Core 2.26 GHz Processor with 8 MB cache (HyperThreading OFF)
2. 72 Gbyte RAM
3. 10 Gbit eth interface to the 44 computing nodes

44 Computing Nodes:

1. 2x Intel® Xeon® E5520 Nehalem Quad Core 2.26 GHz Processor with 8 MB cache (HyperThreading ON)
2. 24 Gbyte RAM

Each Nehalem Quad core CPU has 4 physical cores with SSE extension. Each node has a 2×36.256 GFLOP/sec peak performance calculated by the following formula:

$FLOPS = 4\,\text{cores} \times 2.266\text{GHz} \times 2\,(\text{SIMD double prec.}) \times 2\,(\text{MUL, ADD})$
$= 36.256$ GFLOP/sec.

There are 44 computing nodes which contain 88 physical CPU. The total number of physical cores are 352 (4×88). With hyper-threading the number of cores can be doubled to 704 virtual core. The peak performance of the ATLAS Computing Cluster is $36.256 \times 2 \times 44 = 3190.528$ GFLOP/sec. With full performance ATLAS takes 12.6 kW, 34.2 A, and cosFI= 0.95.

Generating 100 000 integers in one computing node took approximately 0.175 sec. If the number of threads is less than the number of dimensions then the multithreaded running is obvious; every thread checks whether $(k + \gamma) \cdot \Upsilon[i] < \varepsilon$ for all $i < n$ where n denotes the dimension. ATLAS has 44 different nodes which are much more than the number of dimensions in our particular case. If one wanted to use all of the cores then the best way would be to run 44 copies of `MAFRA` in each node. In this case each node should start from different starting points. Generating 44 different appropriate starting points for each copy of `MAFRA` can be done very effectively with the LLL algorithm.

By using `MAFRA` in accordance with LLL it is possible to generate 4.4 million integers within $0.175 + \delta$ seconds where δ is the generating time of the 44 starting points not

exceeding 5000 ms. With the ATLAS Computing Cluster calculating exactly one billion integers that satisfy (2.29) took approximately 39.7 seconds.

Remark 2.8. Generating the 44 integers as starting points with LLL can be done very effectively, however we would like to emphasize that `MAFRA` overperforms LLL in generating many solutions (e.g. one billion) .

2.6.2.3 Test – ATI Radeon 7970 GPU

The third test environment was a Sapphire Vapor-X ATI Radeon 7970 6GB GDDR5 GHz Edition GPU card. Modern graphic cards can be other promising solutions for solving high performance computations. Clearly, in order to implement another fast method for our Diophantine approximation problem one has to take into consideration the usage of GPU cards. In our case the multithreaded version of `FRA` and `AFRA` were implemented for the GPU. In the first step, however, we faced the following problem: there were not any fast *quadruple* precision packages for the GPU. Although some similar packages for the older GPU cards were found written by Andrew Thall [54] and Eric Bainville, these packages were found to be inappropriate to solve our particular challenge. The problem with the package written by Andrew Thall is that it uses too much branching and function calling in the program which costs a lot clock cycles. It comes from the behaviour of the graphical processing unit which evaluates both the `if` and the `else` part of the conditional, and after the computation it uses that data where the logical value was TRUE. `FRA` and `AFRA` contain a lot of logical evaluation, so the usage of this package was not convenient for our purposes. The other package, which was written by Eric Bainville, is faster, but it is for fixed point numbers which was inappropriate, as well.

In conclusion, we developed our own multiplication, addition, subtraction and truncation methods. We applied the Karatsuba multiplication algorithm and some bitwise tricks for the addition and truncation methods. In spite of all these one can observe a huge performance drop-down using the `AFRA` algorithm on the GPU without the usage of the LLL algorithm. After examining `AFRA` we can state that the main problem with this "linear" algorithm is that it was not possible to distribute enough threads on the GPU. Consider for example our 10-dimensional case. One had to add the 1024 integers to the partial results and then multiply them with the irrationals. The problem with this solution is that in the quadruple–adder kernel it was not possible to send in enough threads lowering or hiding the latency. In our case the global work size was twice as big as the local work size, which lead to the performance drop-down. In order to avoid the big performance drop-down we utilized every thread on the GPU just like in the ATLAS Super Cluster. For example, if we wanted to use 2048 threads on the GPU then we

have to generate 2048 different starting points with the LLL algorithm to feed all the threads on the GPU. We also modified the number representation in order to achieve higher speed on this architecture. In that particular case our measurements show that generating 100 000 different integers on the 7970 GPU is 4 times faster than on the CPU.

Combination of the CPU version of LLL and the GPU version of `MAFRA` turned out to be a very effective way to solve simultaneous Diophantine approximation problems.

Chapter 3

The RS-PEAK Algorithm

In this Chapter we present the `RS-PEAK` algorithm for finding large candidates of the Riemann zeta function on the critical line. The algorithm has three main parts:

- Part I - Fast Diophantine Approximation,
- Part II - Prefiltering,
- Part III - Main filtering.

The first part is for generating candidates where large $Z(t)$ is likely by solving simultaneous Diophantine approximations. The second and third part are sieving methods for eliminating weak candidates. In this Chapter we introduce a very special function $F(t)$ which shows in some aspect similar characteristics to $Z(t)$ but is easier to compute.

3.1 Part I - Fast Diophantine Approximation

In 1979 Richard P. Brent noted that unusual large values of $Z(t)$ had been observed where the first few terms in the main sum of the Riemann-Siegel formula have all the same sign (e.g.: the first 72 terms in $Z(30694257.76)$ are all positive [13]). In the main sum of the Riemann-Siegel formula (1.38) one can observe that $\lim_{n\to\infty} \frac{1}{\sqrt{n}} = 0$, so the initial segment of the sum dominates.

Large values of $Z(t)$ are very likely where $\cos(\theta(t) - t\cdot\log n)$ is close to 1 or -1 for many n. Clearly, the largest $Z(t)$ can occur when $\cos(\theta(t) - t\cdot\log n) \approx 1$ or $\cos(\theta(t) - t\cdot\log n) \approx -1$ for all n. In 1989 Odlyzko presented a method for predicting large values of $Z(t)$. "We need to find a t for which there exist integers $m_1, \ldots, m_n$ such that each of $t\log p_k - 2\pi m_k$ is small $(1 \le k \le n)$" [37]. In 2004 Kotnik noted [38] that for $t = \frac{2k\pi}{\log 2}$ we have

$$\cos\left(\theta(t) - \frac{2k\pi}{\log 2}\cdot\log n\right). \tag{3.1}$$

In this case for $n = 1$ we have $\cos(\theta(t))$ and for $n = 2^m, m \in \mathbb{N}$ we have $\cos(\theta(t) - 2km\pi) = \cos(\theta(t))$, so these values have the same sign and reinforce each other in the main summand. The plan is to approximate (3.1) well with $\cos(\theta(t))$ for as many n as possible. Let $\{p_1, p_2, \dots p_s\}$ be a finite subset of primes and $k \in \mathbb{N}$ such that $k\frac{\log p_i}{\log 2}$ are "close to an integer" for all $1 < i \leq s$. Then

$$k\frac{\log p_i}{\log 2} \approx r_i \in \mathbb{N} \tag{3.2}$$

therefore

$$\frac{2k\pi}{\log 2} \approx \frac{2r_i\pi}{\log p_i}\,. \tag{3.3}$$

Hence

$$\cos(\theta(t) - \frac{2r_i\pi}{\log p_i}\log n) \approx \cos(\theta(t))\,, \tag{3.4}$$

for all $n = p_i^m$ $(m \in \mathbb{N})$. Clearly, if one chooses $k \in \mathbb{N}$ in a way such that $k \cdot \frac{\log p_i}{\log 2}$ is close to an integer for as many prime p_i as possible then the expression (3.1) will have the same sign for many n. If one chooses $\cos(\theta(t))$ as large as possible (e.g. near to a Gram point) and one chooses an appropriate $k \in \mathbb{N}$ then one can expect large values of $|Z(t)|$.

Approximating (3.1) well with $\cos(\theta(t))$ for as many n as possible is a simultaneous Diophantine approximation problem. Recall one of the challenges from Chapter 2:

$$\Lambda\left(\left\{\frac{\log(p)}{\log(2)}\,, p \text{ prime}\,, 3 \leq p \leq 31\right\}, 0.01\right). \tag{3.5}$$

Note that approximating (3.1) and generating many elements of Λ are equivalent problems and MAFRA can be used to solve these challenges efficiently.

Remark 3.1. The `RS-PEAK` algorithm is using MAFRA instead of LLL to generate many $k \in \mathbb{N}$ candidates where large $Z(\frac{2k\pi}{\log 2})$ values are expected.

3.2 Part II - Prefiltering

Applying the method presented in the previous section one can generate millions of $k \in \mathbb{N}$ that satisfy $\left\|k \cdot \frac{\log p_i}{\log 2}\right\| < \epsilon$. Calculating $Z(\frac{2k\pi}{\log 2})$ is a very expensive task with the original Riemann-Siegel formula, even with the $\mathcal{O}(t^{1/3})$ time complexity algorithm of Hiary.

As we already noted, the initial segment of the main summand of the Riemann-Siegel Formula is dominant. Exploiting this behaviour of $Z(t)$ in 2004 Kotnik presented [38]

the following function

$$Z(t,k) = 2\sum_{n=1}^{N} \frac{1}{\sqrt{n}} \cos(\theta(t) - t \cdot \log n)) \tag{3.6}$$

where N is $\lfloor (\frac{t}{2\pi})^{1/(2+k)} \rfloor$ for $k = 1, 2$. We expect large $Z(t)$ where $Z(t,k)$ is large due to the fact that $\frac{1}{\sqrt{n}}$ is dominant in the beginning of the summand and less relevant for larger n. Since $Z(t)$ can be calculated in $\mathcal{O}(t^{1/3})$ therefore calculating $Z(t,1)$ is not relevant anymore. Calculating $Z(t,2)$ for very large t values even with larger k (e.g. $k = 3, 4, \ldots$) is still very expensive.

Investigating thousands of large values of $Z(t)$ published by other authors (e.g: Odlyzko, Hiary and Kotnik) led us to create the following function:

$$F(t) = \sum_{n=1}^{\lfloor \log t/2\pi \rfloor} \frac{1}{\sqrt{n}} \cos(\theta(t) - t \cdot \log n)) \,. \tag{3.7}$$

Note that the complexity of $F(t)$ is $\mathcal{O}(\log t)$, which is significantly better than the original $Z(t)$ or $Z(t,k)$. The behaviour of $F(t)$ was substantially investigated. It can be concluded that in many cases $Z(t)$ and $F(t)$ have the same behaviour.

Where peak values of $F(t)$ occur we can expect in a high proportion of the cases that there are peak values of $Z(t)$ as well. Of course there are cases where the behaviour of $F(t)$ and $Z(t)$ is different, but in general the method can be used very effectively to eliminate unlikely candidates [55].

Assumption: *For a given $t \in \mathbb{R}$ where $F(t)$ is large we assume that $Z(t)$ is also large in a high proportion of the cases.*

This assumption is based on our experiment research and many empirical results will be presented to support this connection between $F(t)$ and $Z(t)$.

On a modern computer architecture $F(t)$ can be calculated in less than a second even for $t \approx 10^{1000}$ and can be used to calculate $F(t)$ for large t values. Calculating $Z(t)$ for $t > 10^{40}$ is beyond the current computational capacity. Calculating $F(t)$ gives us a lot of useful information about the behaviour of $Z(t)$, such as the expected growth of $Z(t)$ or the magnitude of $Z(t)$.

Table 1 shows the difference between the calculation speed of $Z(t)$ and $F(t)$, denoted by $\Omega_{Z(t)}$ and $\Omega_{F(t)}$, respectively. For testing purposes we used a Supermicro server equipped with 2 Intel® Xeon® Processor E5-2650 v4 CPU. For calculating $Z(t)$ the $\mathcal{O}(t^{1/3})$ time

complexity algorithm of Hiary was used[1]. $F(t)$ is very simple, so we implemented it in the PARI/GP computer algebra system.

#	t	$Z(t)$	$F(t)$	$\Omega_{Z(t)}$	$\Omega_{F(t)}$
1	69903941711014013853520029.49	3794.501	13.382	$5434s$	$< 1ms$
2	13220924021248300985543923 73.32	-5012.013	-13.841	$17478s$	$< 1ms$
3	59645000709170125023347448 33.72	-4619.42	-14.007	$31753s$	$< 1ms$
4	72146956267479779799849851 46.68	6089.99	14.007	$34194s$	$< 1ms$
5	31616488911549318255796390329.65	-7135.605	-13.467	$60561s$	$< 1ms$
6	10^{100}	N/A	0.059	N/A	$3ms$
7	10^{340}	N/A	1.720	N/A	$31ms$
8	10^{1000}	N/A	0.07	N/A	$824ms$

TABLE 3.1: Computation time of $Z(t)$ and $F(t)$ for some values

3.2.1 Find largest

In order to verify the strength of $F(t)$ various tests are made on earlier published $Z(t)$ values. Let us consider the following simple maximum searching algorithm:

Algorithm 1 `FindLargest`(t, N, Δ)

```
1: t_max, F_max ← 0
2: while t < t + N do
3:     f ← abs(F(t))
4:     if f > F_max then
5:         t_max ← t
6:         F_max ← f
7:     end if
8:     t ← t + Δ
9: end while
10: return (t_max, F_max)
```

One can observe that the most expensive computations in the main summand of $F(t)$ are calculating the square root and the natural logarithm function many times. E.g., for the value $t = 69903941711014013853520000$ the `FindLargest`$(t, 1000, 0.1)$ algorithm calculates $\frac{1}{\sqrt{n}}$ and $\log n$ every time when $F(t)$ is called. In our particular case, $F(t)$ is invoked 10 000 times. However, in our experiments the values of t are $0 \leq t \leq 10^{40}$, and since $\lfloor \log(10^{40}/2\pi) \rfloor \approx 90$ one can use a precomputed lookup table for storing the values $\sqrt{n}$ and $\log n$, respectively. Running `FindLargest`$(t, 1000, 0.1)$ without a precomputed

[1]The algorithm can be downloaded from Github https://github.com/jwbober/zetacalc

table took approximately 3.5 seconds in our test environment. Using a precomputed table the same task took approximately 1.5 seconds. The difference is significant.

Let us analyse the `FindLargest` algorithm applying different Δ stepsizes. Let the starting point $t = 69903941711014013853520000$, where we have

#	Δ	N	t_{max}	F_{max}
1	1	50	69903941711014013853520029	11.355
2	0.1	50	69903941711014013853520029.6	13.08
3	0.01	50	69903941711014013853520029.49	13.382

TABLE 3.2: The result of the `FindLargest` algorithm with different stepsizes

The algorithm finds the appropriate t_{max} values in the interval $[t, t + N]$, where the largest $Z(t)$ occur at $t = 69903941711014013853520029.49$ using $\Delta = 0.01$ stepsize. $F(t)$ in the interval $[t, t + 50]$ with $\Delta = 1$ and $\Delta = 0.1$ stepsizes can be seen in Figure 3.1 and 3.2. The difference is striking. One can observe that $F(t)$ is more dense with $\Delta = 0.1$ than with $\Delta = 1$.

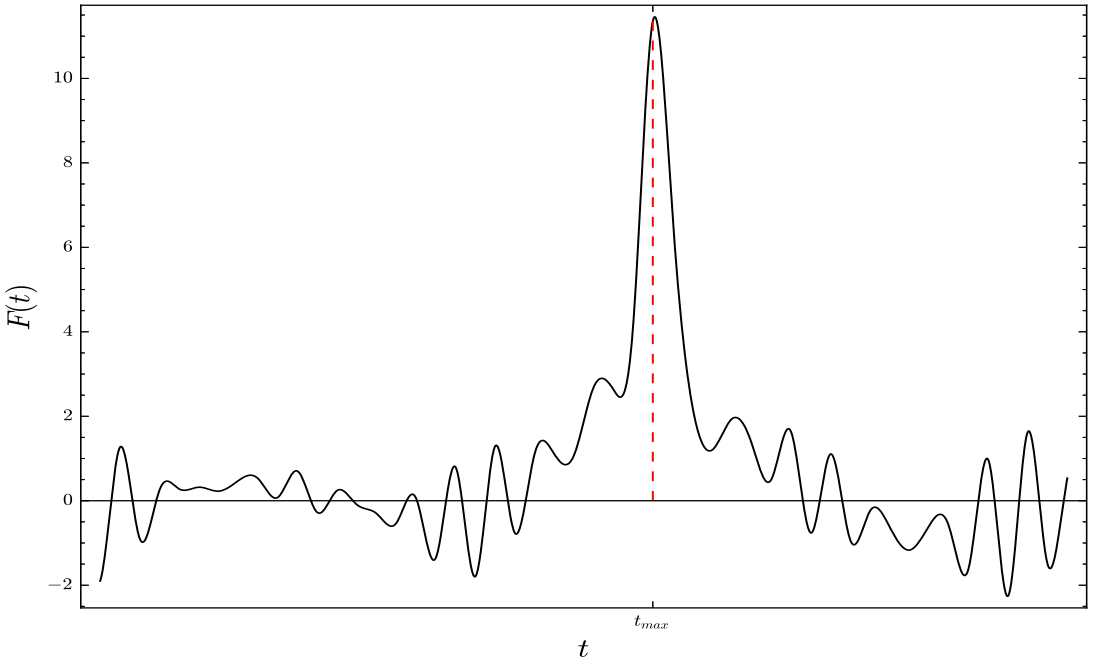

FIGURE 3.1: Plotting $F(t)$ with stepsize $\Delta = 1$

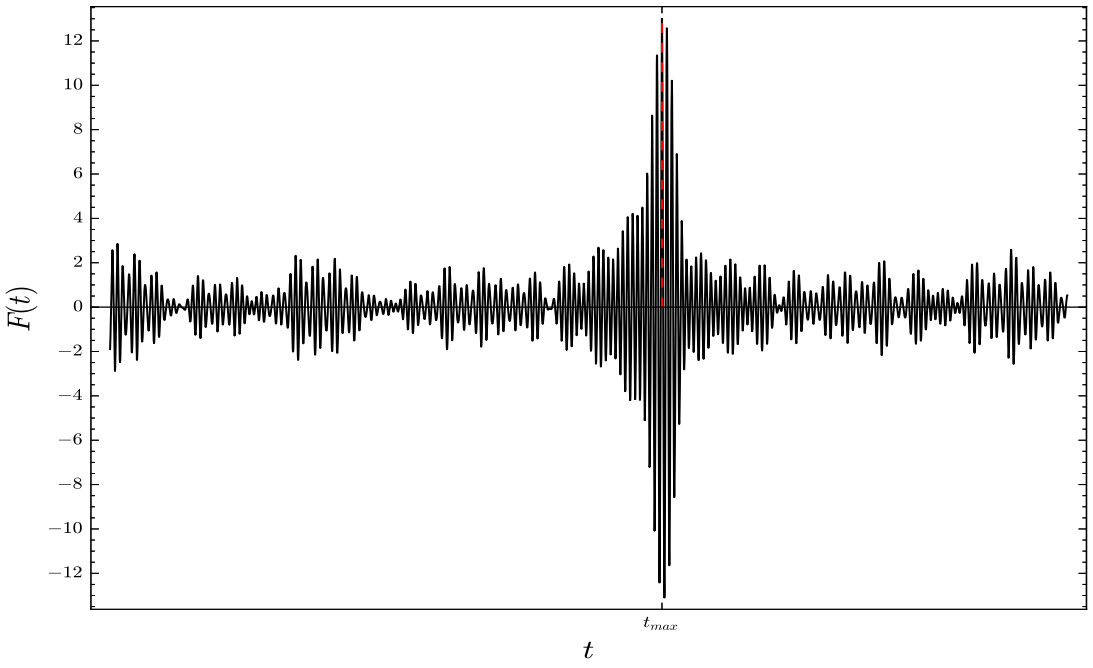

FIGURE 3.2: Plotting $F(t)$ with stepsize $\Delta = 0.1$

One can observe that the real behaviour of $F(t)$ can only be seen with the smaller stepsize $\Delta = 0.1$. Computational experiments show that in the range of interest, the stepsize $\Delta = 1$ is sufficient for finding t_{max} with small deviation.

Calculated $F(t)$ with stepsize $\Delta = 1$ and stepsize $\Delta = 0.1$ can be seen in Figure (3.3).

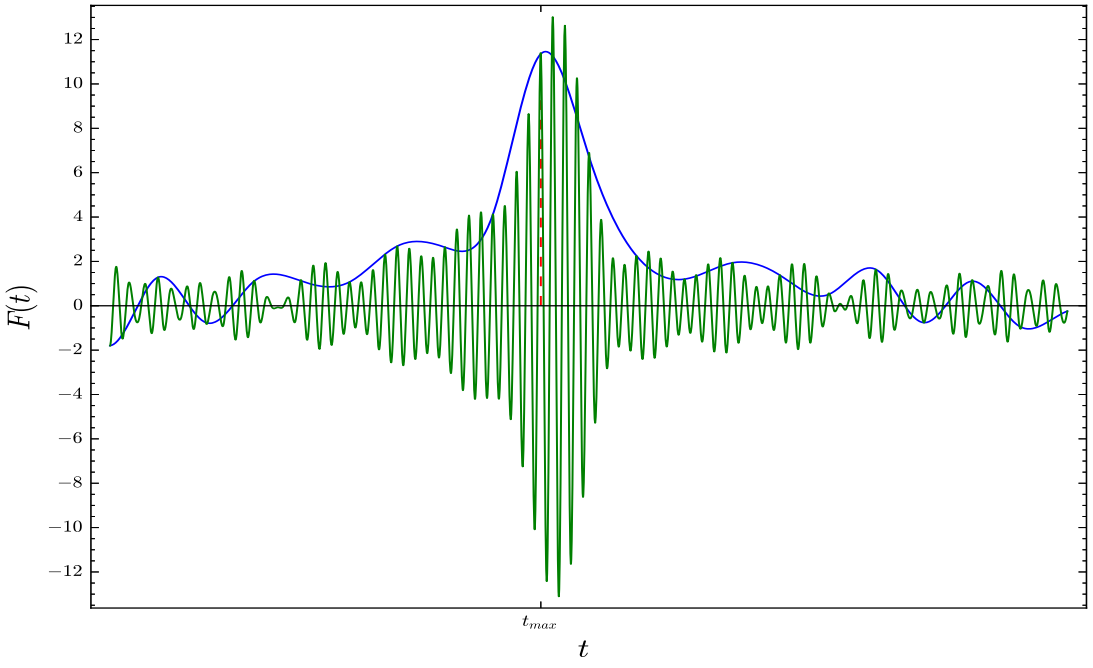

FIGURE 3.3: Plotting $F(t)$ with stepsize $\Delta = 1$ and stepsize $\Delta = 0.1$

3.2.2 Deviation of $F(t)$ and $Z(t)$

In many cases $F(t)$ can be used effectively to indicate where large values of $|Z(t)|$ are likely [60]. We analysed the `FindLargest` algorithm for many different t values with different N and Δ parameters. Let $Z_{max} = Z_{max}(t_0, N)$ denote such a value t where $|Z(t)|$ is the largest in the interval $[t_0, t_0 + N]$. Similarly, let $F_{max} = F_{max}(t_0, N)$ denote such a value t where $|F(t)|$ is the largest in the interval $[t_0, t_0 + N]$. In order to measure the strength of the $F(t)$ function we are interested in the deviation of $Z_{max}(t_0, N)$ and $F_{max}(t_0, N)$. Let us define the function σ as

$$\sigma = \sigma(t_0, N) = 100\frac{|Z(Z_{max}(t_0, N)) - Z(F_{max}(t_0, N))|}{Z(Z_{max}(t_0, N))}$$

Table 3.3 displays the output of the `FindLargest`(t, N, Δ) algorithm (t values found by `RS-PEAK`) for different parameters together with σ.

$t_0 = 356071078353654500$					
$Z(Z_{max})$	N	Δ	Z_{max}	F_{max}	σ
1287.14	100	0.01	$t_0 + 62.22$	$t_0 + 62.22$	0%
$t_0 = 6578787583549202400$					
$Z(Z_{max})$	N	Δ	Z_{max}	F_{max}	σ
−1368.459	100	0.01	$t_0 + 0.03$	$t_0 + 0.03$	0%
$t_0 = 1322092402124830098554392000$					
$Z(Z_{max})$	N	Δ	Z_{max}	F_{max}	σ
−5012.013	1000	0.01	$t_0 + 373.32$	$t_0 + 373.32$	0%
$t_0 = 3161648891154931825579639000$					
$Z(Z_{max})$	N	Δ	Z_{max}	F_{max}	σ
−7135.606	1000	0.01	$t_0 + 329.65$	$t_0 + 329.66$	0.46%

TABLE 3.3: Output of the `FindLargest`(t, N, Δ) algorithm for different t values

We also tested our $F(t)$ function on different values published by other authors. Figures (3.5)-(3.8) show the neighbourhood of F_{max} for large values of t published by Hiary[2]. In each case the appropriate F_{max} values differ only with $\sigma < 0.5\%$.

[2] https://people.math.osu.edu/hiary.1/

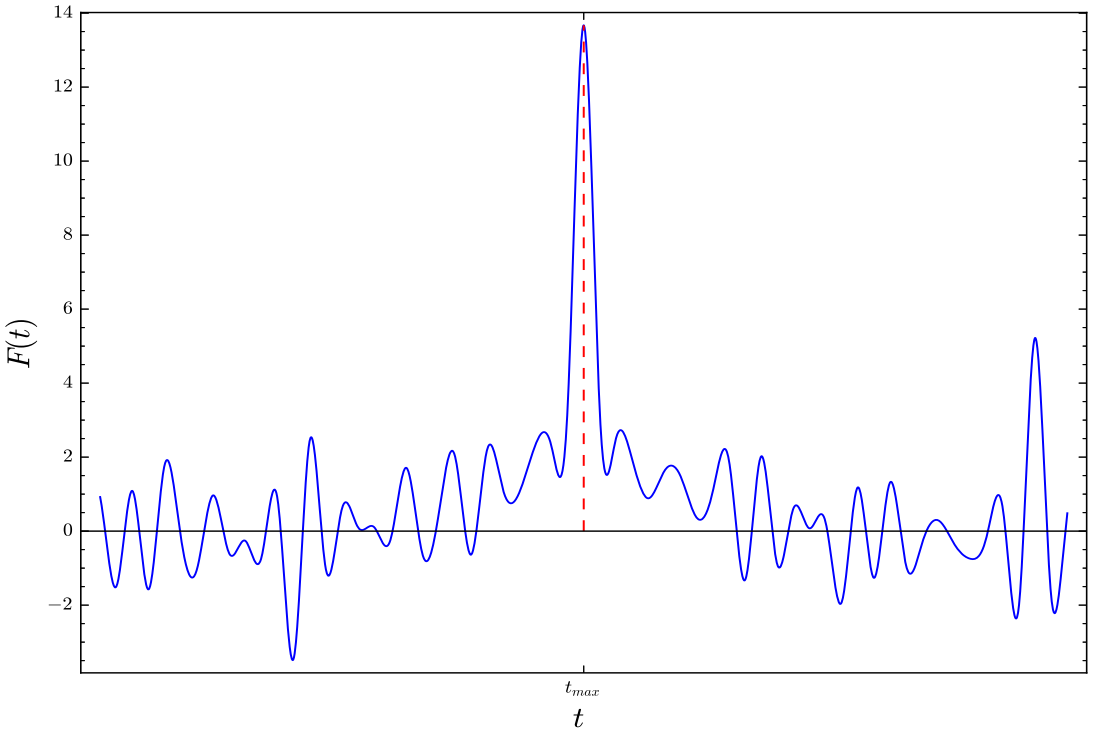

FIGURE 3.4: $Z(t) \approx 16244.86, F(t) \approx 13.67$

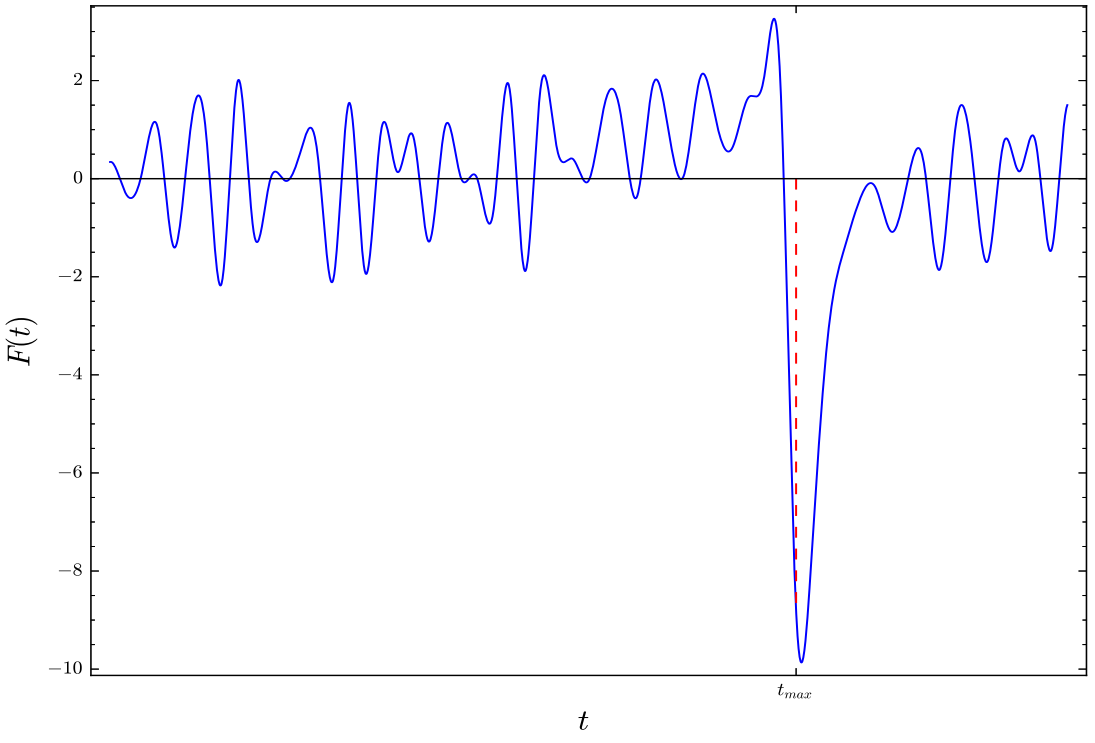

FIGURE 3.5: $Z(t) \approx -14055.89, F(t) \approx -14.218$

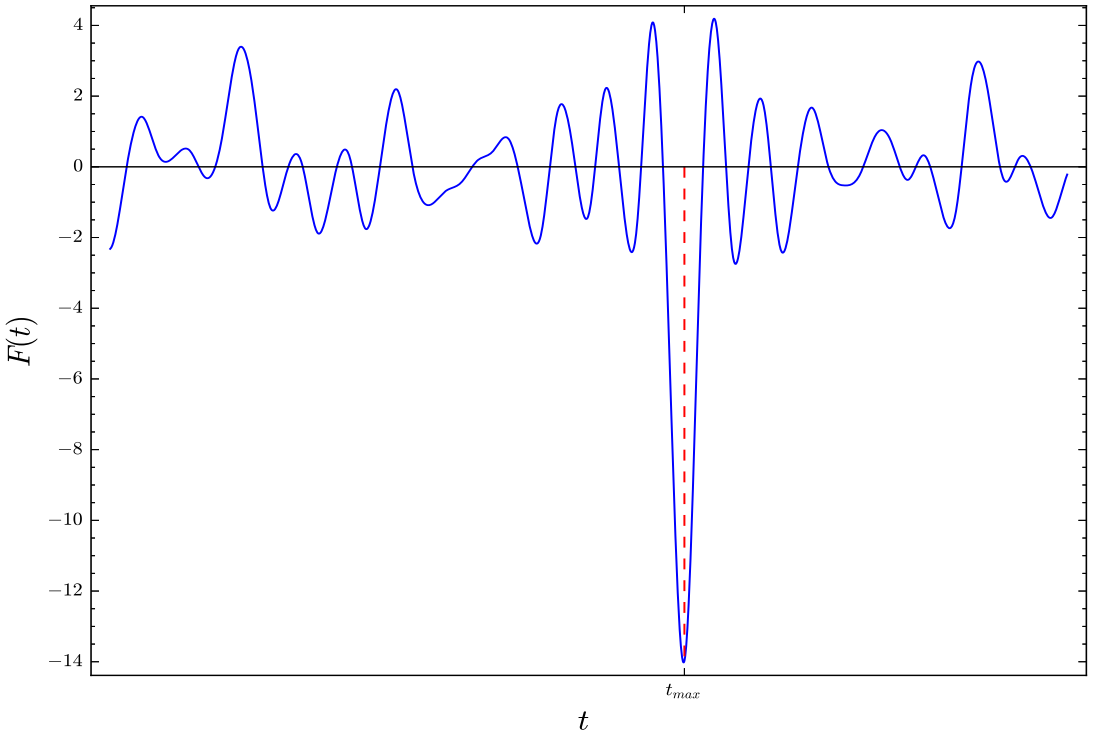

FIGURE 3.6: $Z(t) \approx -13558.833, F(t) \approx -14.242$

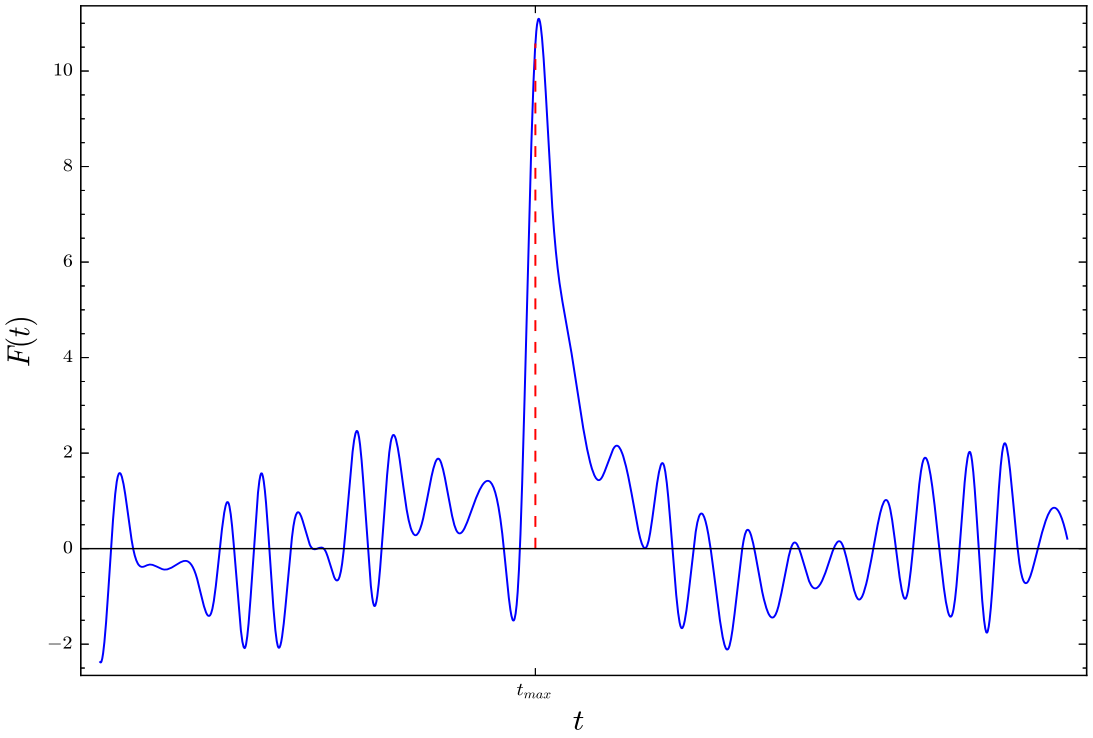

FIGURE 3.7: $Z(t) \approx 13338, F(t) \approx 11.430$

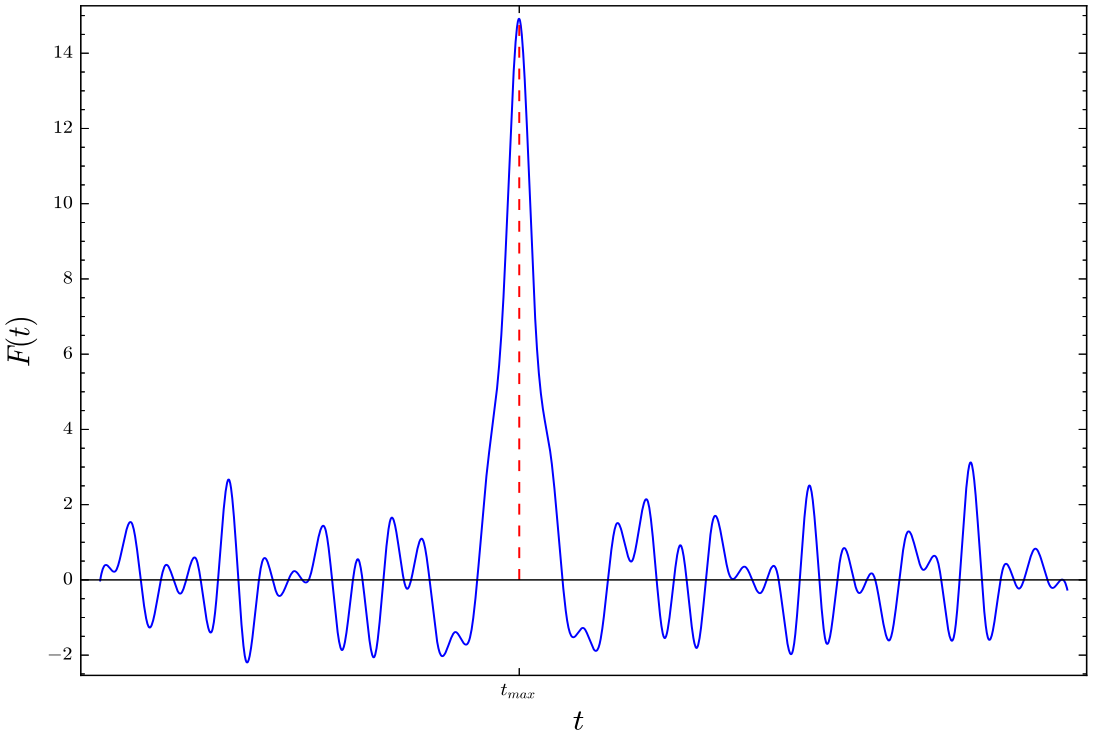

FIGURE 3.8: $Z(t) \approx 12021.094, F(t) \approx 14.807$

These (and many other similar) tests suggest that $F(t)$ can be used effectively to indicate where large $Z(t)$ is likely. Investigating more than 1000 large values in different intervals and different heights suggests that $F(t)$ is effective for finding candidates where large $Z(t)$ is likely.

Remark 3.2. It can be concluded that $F(t)$ can be used for eliminating the unlikely candidates generated by MAFRA.

3.3 Part III - Main filtering

It is easy to see that the order of $F(t)$ depends on the order of t. Calculating thousands of $F(t)$ for different t values led us to create the following function:

$$A(t, B_1, B_2) = \sum_{n=B_1}^{B_2} \frac{1}{\sqrt{n}} \cos(\theta(t) - t \cdot \log n)\,. \tag{3.8}$$

In practice, for eliminating weak candidates, the values $A(t, 1, 1) = \cos(\theta(t))$, $A(t, 1, 100)$, $A(t, 1, 1000)$ and $A(t, 1000000, 1100000)$ were used. More research would be needed to thoroughly analyse thoroughly the best choice of B_1, B_2. These functions can be applied

in the following way: In case of $A(t, B_1, B_2) > M$ one can assume that $Z(t)$ is large. Clearly, M depends only on B_1, B_2 and does not depend on the order of t.

3.4 Combination of parts I, II and III

The algorithm `RS-PEAK` is based on the combination of the three basic strategies. The `RS-PEAK` algorithm is implemented for different architectures.

The pseudocode of the `RS-PEAK` algorithm can be found in the Appendix. In the following section many new results and records are presented that were achieved using `RS-PEAK`.

Chapter 4

Computational Results

The main result of this chapter is to present the computational results of the *Riemann Zeta Search Project*. More than 5 million candidates were found where large $Z(t)$ values are expected during a 4-year period. We present some results using a single desktop and using a complex distributed environment.

4.1 Desktop computation

In this section many different tests are presented for proving the strength of the `RS-PEAK` algorithm. All computations were made on a single Laptop equipped with a 2.90 GHz Intel© Core i7-3520M Ivy Bridge processor with 16 GB RAM. The `RS-PEAK` algorithm was implemented in C++ with an extension of GNU MP 5.1.2.

4.1.1 Searching candidates in the range from $k = 10^{15}$ to 10^{16}

One of the strengths of the `RS-PEAK` algorithm is the ability to locate many large values in a given interval. In this challenge we are focusing on locating as many $Z(t)$ values as possible in the range from $k = 10^{15}$ to 10^{16} where $Z(\frac{2k\pi}{\log 2}) > 500$. The following parameters were used: $n = 9$, $\varepsilon = 0.1$, $C_1 = 0.95$, $C_2 = 10$, $C_3 = 16.3$, $C_4 = 40$ and C_5 was not set[1]. Applying these parameters we found 10775 candidates within 2 hours in a single desktop PC where $Z(t) > 500$ is probable. Calculating every candidate took approximately 1 hour by the ATLAS Super Cluster (2.6.2.2). 10305 $Z(t)$ values were larger than 500. This is a 95.6% success rate. 1133 values were larger than 800. 104 values were larger than 900 and 9 values were larger than 1000 (see TABLE 4.1).

[1]The C parameters are defined in the RS-PEAK algorithm in the following way: $C_1 = A(t, 1, 1) = \cos(\theta(t))$, $C_2 = F(t)$, $C_3 = A(t, 1, 100)$, $C_4 = A(t, 1, 1000)$ and $C5 = A(t, 1000000, 1100000)$

TABLE 4.1: $Z(t) > 1000$

$Z(t)$	t
1007.54	44686099305781486.48
1081.07	54671407423795346.46
−1046.46	71319494106624241.91
−1028.36	71762434186826396.45
1003.02	80587673978410171.15
−1024.21	81704260588233260.79
1032.17	83021350953863240.97
−1007.21	84725216736147134.29
−1008.32	89793576527342239.72

4.1.2 The 1-hour challenge

Find t values in different magnitudes where t is between 10^n and 10^{n+1} for $n = 17, 18, 19, 20, 21, 22, 23$ and $1100 < |Z(t)| < 1300$. `RS-PEAK` found the 7 different values in 47 minutes (see TABLE 4.2).

TABLE 4.2: $1100 < |Z(t)| < 1300$

n	$Z(t)$	t
17	1287.14	356071078353654562.22
18	−1280.45	5641246466760347904.45
19	1291.72	16410490320565230996.92
20	−1230.54	155233938092240166695.72
21	1191.33	9068682153663397254682.71
22	−1156.34	503596603024746753161 27.43
23	1274.58	34850259005960816496 8431.0

`RS-PEAK` can be used to search for $Z(t)$ values at a larger magnitude. Let $a = 2300$ and $b = 2700$. Find 10 different $Z(t)$ values where $a < |Z(t)| < b$ for t from $9.668563 \cdot 10^{22}$ to $9.668568 \cdot 10^{22}$. `RS-PEAK` found 10 different values approximately in 1 hour in this range (see TABLE 4.3).

TABLE 4.3: $2300 < |Z(t)| < 2700$

$Z(t)$	t
-2624.76	96685655632550213096463.749
2516.94	96685673426064241734832.767
2550.92	96685665029898136813234.12
2538.56	96685642632814140374765.25
2435.50	96685652808323950868784.45
-2420.08	96685635339779103874796.19
2402.16	96685648060184894683096.33
-2379.7	966856714026975528678 50.962
2381.35	96685671761733939382513.69
2343.92	96685635552106721386439.15

4.1.3 Large $\varphi(t)$ values

It is known that $\zeta(s)$ is unbounded on the critical line, so $|Z(t)|$ can be arbitrarily large as t goes to infinity. It is known as well (see equation (1.25)) that the zeros of the zeta function become more and more dense as one goes upwards in the critical strip. Hence, one can find $|Z(t)| > 1000$ more likely in the region around 10^{1000} than around 10^{20}. Consider the following two $Z(t)$ values:

$$\begin{aligned} Z(t_1 = 735152777740986806730805.8) &= -1144.41 \\ Z(t_2 = 54671407423795346.46) &= 1081.07\,. \end{aligned}$$

Although $|Z(t_1)| > |Z(t_2)|$ the point t_1 is much larger than t_2.

Let us define the function

$$\varphi(t) = \frac{\log|Z(t)|}{\log(t)}. \tag{4.1}$$

In this particular example $|Z(t_1)| > |Z(t_2)|$, although $\varphi(t_2) > \varphi(t_1)$.

Bourgain showed [64] that

$$\zeta\left(\frac{1}{2} + it\right) = \mathcal{O}(t^{13/84+\epsilon}) \tag{4.2}$$

for every $\epsilon > 0$, and assuming the Riemann hypothesis

$$\lim_{t\to\infty} \varphi(t) = 0. \tag{4.3}$$

The largest known $\varphi(t)$ found by Odlyzko in 1989. For $t = 50328687692882891111.35$ we have $Z(t) \approx 1581.7$ and $\varphi(t) \approx 0.17106$. This record can be broken easily applying the `RS-PEAK` algorithm. After searching in the same region for 5 hours we found $Z(t) \approx -1634.1$ and $\varphi(t) \approx 0.17127$ for $t = 57662612013338230 98.71$. Within 10 hours approximately 1000 values were found where $\varphi(t) > 13/84$. Moreover, one can find values where $\varphi(t) > 0.18$ and $Z(t) > 1000$, e.g for $t = 54671407423795346.461$ we have $Z(t) \approx 1081.16$ and $\varphi \approx 0.18126$.

It has turned out that `RS-PEAK` can be used efficiently to find the upper bound of t values for different C constants where $\varphi(t) > C$. E.g. for $t = 6436526919750171929565.992$ we have $Z(t) = -2942.71$ and $\varphi(t) \approx 0.15905$. To the best of our knowledge, there is no larger t known where $\varphi(t) > 13/84$ occurs.

Using the `RS-PEAK` algorithm in a distributed way would be a very promising method for searching for very large $Z(t)$ values. In the next section we present the distributed computing results of the algorithm.

4.2 Distributed computing – The Riemann Zeta Search Project

It was shown that `RS-PEAK` is very efficient even on a single Desktop PC, so applying `RS-PEAK` in a distributed environment promises to find even more and larger candidates.

SZTAKI Desktop Grid (SzDG) is a BOINC project located in Hungary run by the Computer and Automation Research Institute (SZTAKI) of the Hungarian Academy of Sciences. The Zeta-search application is a multithreaded `RS-PEAK` algorithm written by the author of this PhD thesis. The application has been deployed on the SZTAKI Desktop Grid in order to find many t candidates where large $Z(t)$ is likely.

In a distributed way, thousands of computer from all over the world were searching large $Z(t)$ candidates applying the `RS-PEAK` algorithm. The name of the project is the *Riemann Zeta Search Project* established by the author of this PhD thesis. The multithreaded `RS-PEAK` algorithm was running on the SZTAKI Desktop Grid from November 2013 till November 2017.

4.2.1 Overview

This subsection is going to introduce the computational environment of the SZTAKI Desktop Grid .

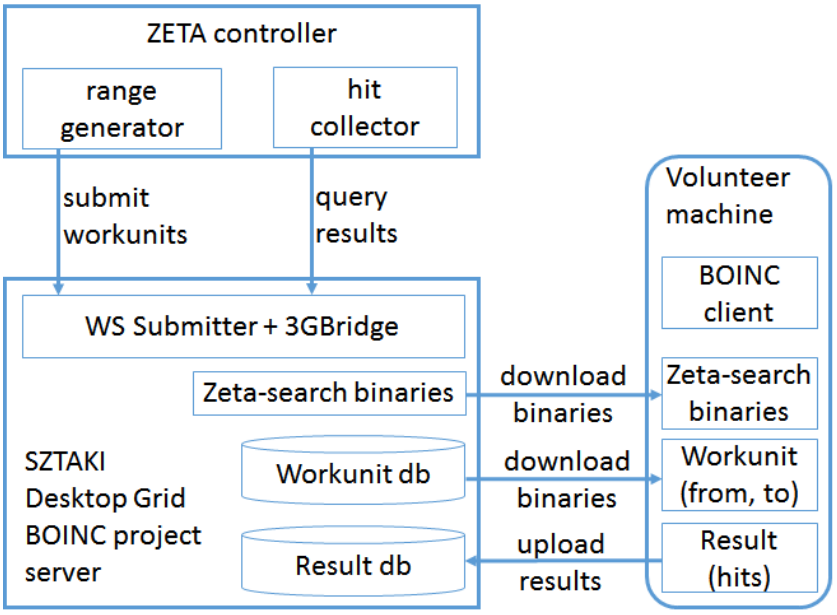

FIGURE 4.1: Overview of the SZTAKI Desktop Grid infrastructure for Zeta calculation [66]

The key components of the overall infrastructure (see Figure 4.1) are the followings:

1. Zeta-search application binaries implementing the `RS-PEAK` algorithm,
2. BOINC server coordinating the distribution of millions of workunits to BOINC clients,
3. BOINC client for executing Zeta-search workunits on volunteers' machines,
4. 3GBridge for generating workunits,
5. Zeta controller for generating parameters.

The Zeta-search application has been developed to implement the `RS-PEAK` algorithm. Parameters for this application include the range (minimum and maximum number of t values among which the search to be performed) or list of ranges and a threshold value which represents the minimal value above which the t value is considered as a candidate. The application has been integrated to the BOINC [65] environment to make it executable under the control of the BOINC client, running on the volunteers' machines.

BOINC together with the extensions from the SZDG package [66] performs the distribution of workunits (jobs) to the volunteers' machines. In BOINC a volunteer person can join to a particular project by registering an account on the project homepage. The BOINC client is a piece of software that is able to communicate to the BOINC server to get workunits, to execute the workunit and to upload the result. After downloading, installing and running the BOINC client on the volunteer's machine, one can attach his/her BOINC client to a BOINC project with the given credentials (email/password). After attaching to a project it starts fetching workunits containing a pack of input files and parameters for a certain application.

The BOINC client downloads the application binary referred by the workunit. Then the BOINC client starts executing the application in the background with the input files and parameters described in the workunit. Once the workunit finishes, the results are uploaded and reported to the BOINC project and new workunits are fetched. BOINC ensures the distribution of millions of workunits by the server towards the attached volunteers.

The SZTAKI Desktop Grid [66] software package is an extension for the BOINC software on the field of job submission, execution and API. Its main contribution is the 3GBridge component – used in this solution – performing the creation of workunits inside the BOINC database. Once a job (application with arguments, input and output files, etc.)

has been submitted through the WS Submitter into 3GBridge, a new BOINC workunit is generated. The status can be queried and the results can be retrieved. WS Submitter is the web-service endpoint for the 3GBridge (see Figure 4.1).

SZTAKI Desktop Grid is not only a software package, but also a BOINC based volunteer desktop grid project operated by the Laboratory of Parallel and Distributed Systems in the Institute for Computer Science and Control of the Hungarian Academy of Sciences, Budapest, Hungary. The SZTAKI Desktop Grid BOINC project is hosted on a virtual machine on the cloud computing infrastructure of the laboratory. The virtual machine has Debian GNU/Linux 6.0.7 (Squeeze) OS with 4 core AMD64 CPU (KVM). Due to low resource requirements of BOINC, the server runs with 2GB RAM. For the workunits to store the server has 500GB. The bandwidth of its network connection is 1Gbit/s.

The SZTAKI Desktop Grid Project was established in 2005 and has currently about 40 thousand volunteers and 112 thousand hosts registered. However, the number of actively working hosts is much lower since hosts and users join and leave frequently in a BOINC environment. The number of active hosts is approximately 2200 owned by about 1700 active users. A host is considered active if at least one finished workunit is reported by the host within 48 hours.

In the infrastructure the Zeta controller (see Figure 4.1) is responsible for generating the load (i.e. millions of jobs) for the BOINC project. Range generator is performing the calculation of the ranges to be processed and submits the corresponding jobs to SZDG through the WS Submitter. The Hit collector downloads the outcome of Zeta-search executions and extracts the valuable results, i.e. the value of t and $|Z(t)|$.

Overall, the flow of jobs starts with the Zeta-controller component by a range generator, continues with 3GBridge and BOINC and ends up on the volunteer machines which downloads the Zeta-search executables and parameters. Once the workunit has been processed, the results travel back through the same route to the Zeta controller server (see Figure 4.1) machine.

4.2.2 Distribution of ranges among multiple BOINC projects

The Zeta search controller server is able to feed multiple BOINC projects with jobs (ranges) as it is depicted in Figure 4.2. When the capacity of a BOINC project is not enough for processing the workunits fast enough it is needed to attach further resources. The Parallel and Distributed Systems Laboratory operated multiple BOINC projects for some years. In order to speed up the computation for the Zeta-search application, the Zeta controller machine was configured to submit jobs to multiple BOINC projects

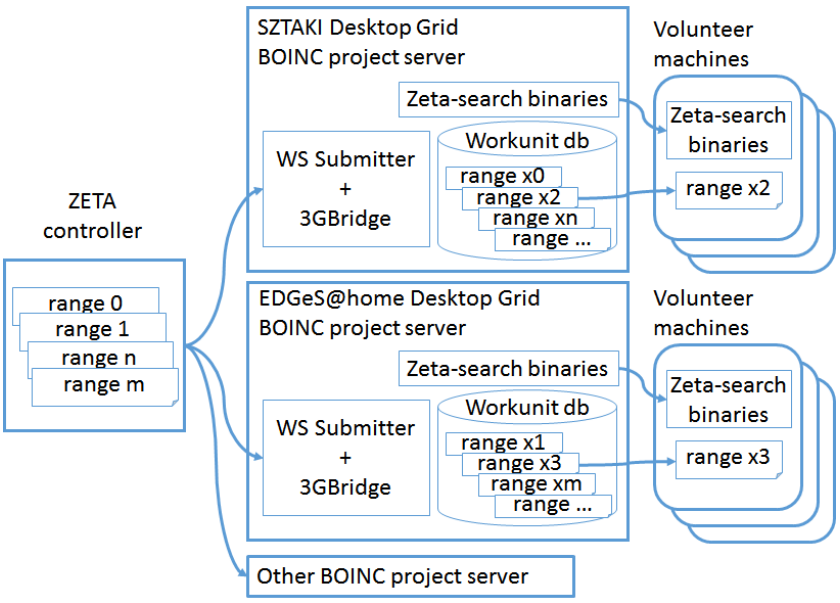

FIGURE 4.2: Distribution of ranges among BOINC infrastructures

through the WS Submitter and 3GBridge components. ZETA controller was responsible to keep track of the different ranges submitted to the different projects and to detect if the computation for a certain range has not arrived in time from one of the BOINC projects.

Until the end of 2015, the laboratory has assigned the EDGeS@home BOINC project resources (volunteers) for executing Zeta-search calculations. During that period both BOINC projects issued workunits with the same applications but with different parameters (ranges). The accumulated performance increased the processing speed by 15-20 percentage since the volunteers on the SZTAKI Desktop Grid was 5-6 times more than on the EDGeS@home BOINC project.

Now, this architecture gives a justification of creating the Zeta controller job (range) generator as a separate component instead of developing it as an integrated component inside the BOINC architecture. With this solution it was possible to assign multiple BOINC projects for the same application. Moreover, 3GBridge is able to create jobs not only for BOINC, but for other systems (due to its pluggable architecture) therefore other type of resources could have been attached. Due to the huge computational capacity requirement of the Zeta calculation the cheapest, i.e. BOINC resources were used.

4.2.3 Architecture of the Zeta controller

One of the key components of the execution environment is the Zeta controller. The controller has several different functionalities:

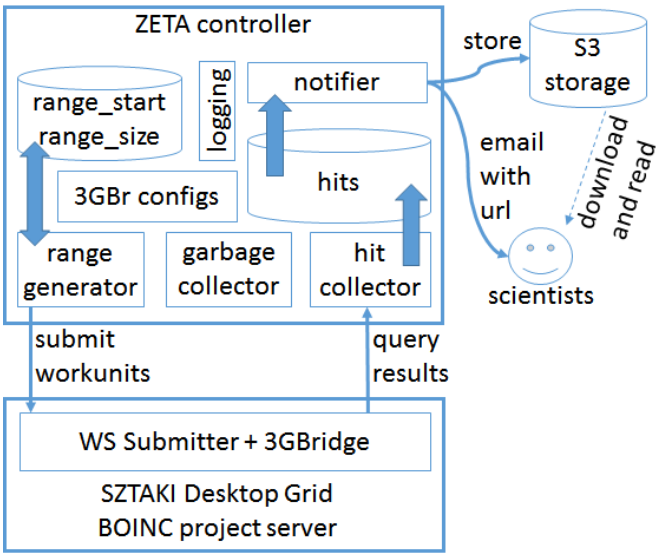

FIGURE 4.3: Architecture of the Zeta controller server

1. Generates the ranges for which jobs can be created and processed as workunits on volunteer machines,
2. Makes sure the numbers to be scanned through are covered by the ranges,
3. Detects if result has not arrived back for a certain range,
4. Resubmits failed or lost workunits,
5. Distributes the generated ranges among the BOINC projects,
6. Keeps the number of jobs in the BOINC projects on a predefined level,
7. Collects results and stores the value of t and $|Z(t)|$ in the database,
8. For a predefined interval (currently one week) it summarises hits, creates and stores a report on an external storage and notifies scientists,
9. Removes unnecessary entries from the database.

Range generator (see in Figure 4.3) is invoked periodically, i.e. once per hour. First it queries the number of unsent workunits on the target BOINC project to detect the available freely downloadable workunits for the clients. If this number is close to zero the frequency or the number of maximum unsent workunit must be increased. Once it has this number then it generates the number of ranges needed to fill up the number of unsent workunits to a predefined level. This level can be specified per BOINC project. With this mechanism the workunit generation can follow the changing capacity of the volunteer computational systems.

The same range generation mechanism is invoked for each configured BOINC project (see 3GBr configs in Figure 4.3) with their own configuration parameters. The range generator reads the actual starting number from the database and updates it when new ranges have been generated. The size of the range is also a configuration parameter which is proportional with the amount of work a workunit contains (see range_start, range_size in Figure 4.3).

The Hit collector (see in Figure 4.3) is also invoked periodically, i.e. once per hour. First, it queries all the finished jobs stored in 3GBridge of the BOINC project and downloads their outputs. It then extracts the results and stores them in the database (see hits in Figure 4.3). Moreover, this component also queries and stores the amount of computational capacity spent for processing the given range. It is needed for summarising the overall computational capacity used for finding candidates.

The Notifier (see in Figure 4.3) is also executed periodically, i.e. once per week in order to summarise the large Zeta values found during the last week. This report is generated and uploaded to a storage accessible for the scientists. Then the scientists are notified via email containing the url of the report.

The operation of all the components are followed by using a logging subsystem (see in Figure 4.3) to ease finding problems if any. Garbage collector is originally designed to cleanup hits and already reported values, but it is not used currently, since hits do not consume large amount of storage space.

The careful and modular design of the zeta controller attached to the SZTAKI BOINC projects resulted in continuous operation of the Zeta controller for 4 years without interruption caused by malfunction during the generation of almost 34 million jobs (ranges).

4.3 Main achievements

Approximately 30 000–50 000 candidates were found weekly by the SZTAKI Desktop Grid applying the `RS-PEAK` algorithm.

During the 4-year period 5 597 001 candidates were found where large $Z(t) > 1\,000$ are expected. The values of most promising candidates were calculated by the ATLAS Supercomputing Cluster (2.6.2.2) applying the $\mathcal{O}(t^{1/3})$ algorithm of Hiary[2].

[2]The C++ implementation of the algorithm written by Jonathan Bober and Ghaith Ayesh Hiary is publicly available at `https://github.com/jwbober/zetacalc`

4.3.1 The top 30 largest $Z(t)$ values

The top 30 largest $Z(t)$ values found by the Riemann Zeta Search Project can be seen in TABLE 4.4. All values were verified by the ATLAS computing cluster.

#	t	$Z(t)$	$\varphi(t)$
1	3106788336290839656675405765936 82.05	16874.202	0.130
2	2961571595742212001859888400019 06.62	−16478.028	0.130
3	2860941159455196369121404699191 09.57	16093.351	0.130
4	3327168444384153714981798591382 03.56	15684.406	0.129
5	9066617388021913893282064021886 2.84	15601.620	0.131
6	1278342904333757441436560707065 06.32	−15566.607	0.131
7	2845670144939668837484722465519 6.74	−15484.420	0.133
8	1674954871436369183239459082492 96.67	15328.836	0.130
9	1269450622692969068558500026653 42.19	−15302.703	0.130
10	4211469190866660705813305070264 94.23	−15027.882	0.128
11	3328295538118267545736726061413 52.02	−14954.363	0.128
12	6229499185841081572805092580489.41	−14335.948	0.135
13	1512401940584871461445705753841 08.86	−14244.786	0.129
14	1913781731100123329784590907991 0.84	−14186.433	0.133
15	6080550187116167583688613601408 2.64	14110.835	0.131
16	8265759880803068655420497930087 5.40	14024.777	0.130
17	1970307186839852852378315796522 7.7	−13870.969	0.128
18	2341323084314225580406157863680 7.79	13535.628	0.132
19	2012715703531971275423303851120 39.94	−13268.415	0.128
20	1605816737694237992721494486236 13.46	−13127.222	0.128
21	1183680343227839025296224701797 29.85	−13123.661	0.128
22	3187106399672231545789906400009 0.11	12867.952	0.130
23	2980201309574460537774188072646 7.46	12815.395	0.131
24	1937283455248936043838735393809 24.64	−12703.177	0.127
25	6825180535904004649459580050276.95	12561.880	0.133
26	4033389921830351782905024505927 81.48	−12444.603	0.126
27	3352318733050074925244708729890 7.34	−12285.930	0.130
28	1058685097611418455518065283976 1.6	12108.987	0.132
29	9662259498245421254629126899658.04	−12070.186	0.132
30	2230413619455576919824940140320 7.02	11991.750	0.130

TABLE 4.4: Largest $Z(t)$ and $\varphi(t)$ found by the Riemann Zeta Search Project

4.3.2 Largest Z(t) value

On 30^{th} of November, 2015 a very promising candidate k was found by the SZTAKI Desktop Grid. For $k = 342734054562393686017347380230 84$ we expected a very large value of $|\zeta(1/2+i\frac{2k\pi}{\log 2})|$. The value was investigated before the expensive $Z(t)$ calculation.

It is easy to verify that

$$\|342734054562393686017347380230 84 \times \frac{\log(p)}{\log(2)}\| < 0.01 \tag{4.4}$$

for all $3 \leq p \leq 31$, thus,

$$\cos(\theta(t) - \frac{2k\pi}{\log 2} \cdot \log n) \approx \cos(\theta(t)) \tag{4.5}$$

holds for many n.

For $t = \frac{2k\pi}{\log 2}$ we have

$$Z(3106788336290839656675405765936 82.05) \approx 16874.202. \tag{4.6}$$

To the best of our knowledge, at the time of writing this thesis it is the largest $|Z(t)|$ ever calculated. This value was published and presented in the Journal of Grid Computing [55] in 2017. Applying the $\mathcal{O}(t^{1/3})$ algorithm of Hiary the exact value was verified by two independent servers eliminating the possibility of any error in the computation.

Furthermore, we have calculated the exact values of $\zeta(1/2+i(t+T))$ with $-10 < T < +10$ applying $\Delta = 0.01$ stepsizes. The calculated $Z(t)$ values can be seen in Figure (4.4). The calculated $F(t)$ and its neighborhood can be seen in Figure (4.5).

As we already noted in the introduction, in 1979 Brent observed an unusually large $Z(t) \approx 79.6$ near the 70354406^{th} Gram point [13]. One can observe that our t value occurs near the $3559050906060807263882379565422420^{th}$ Gram point.

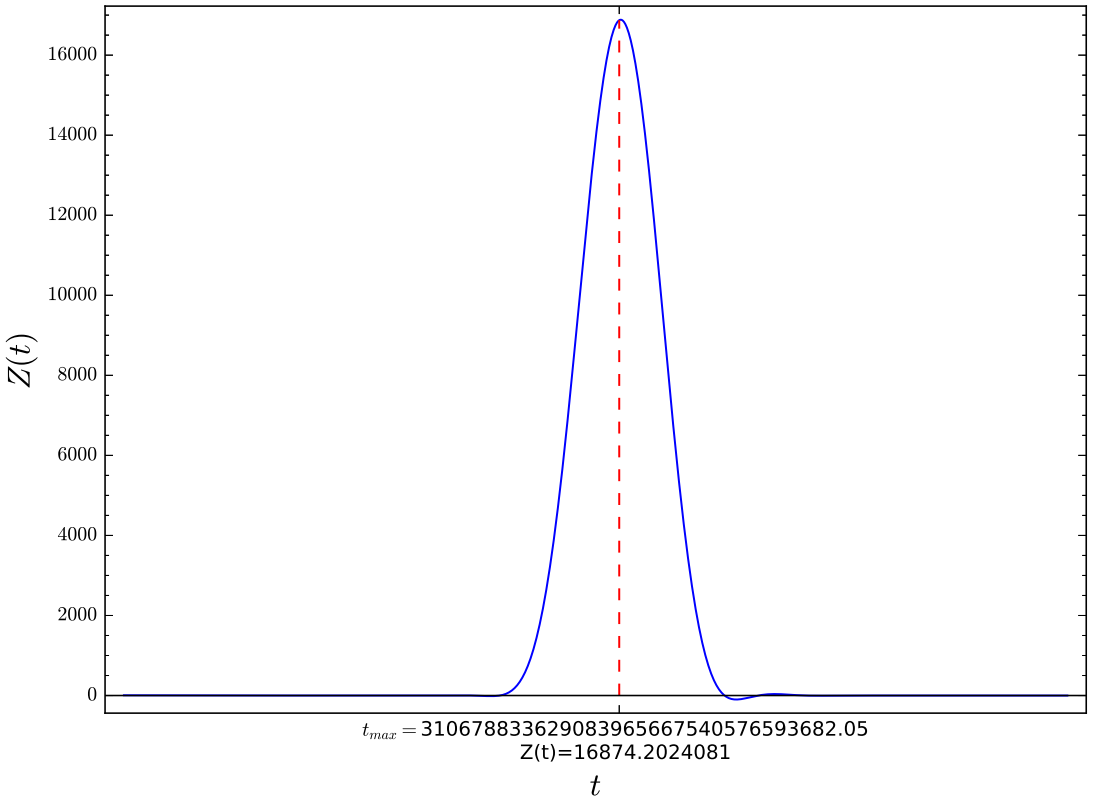

FIGURE 4.4: Largest local maximum with $Z(t) \approx 16874.202$

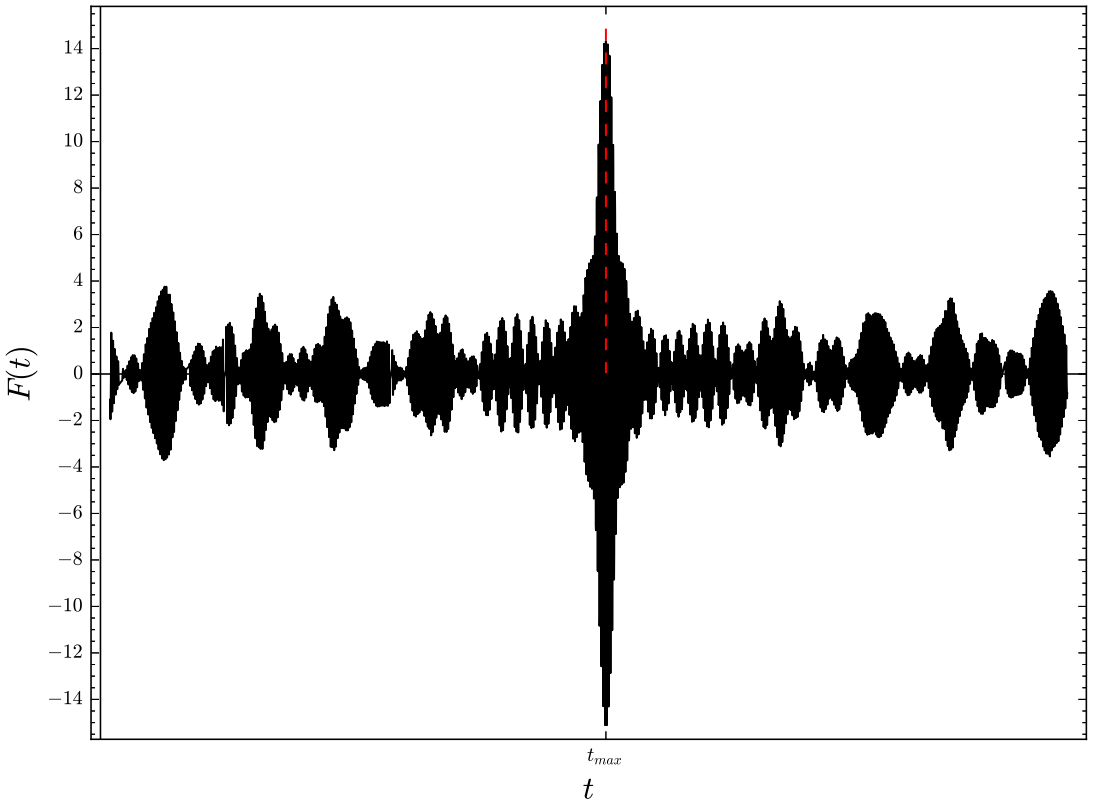

FIGURE 4.5: Peak value of $F(t)$ where the largest $Z(t)$ found occurs (near 3.1067×10^{32})

Chapter 5

Conclusion and future work

During the 4-year period of the Riemann Zeta Search Project millions of $Z(t)$ values were calculated. Large $Z(t)$ values and the $\varphi(t)$ function introduced in (4.1) were substantially investigated. We have collected the largest $\varphi(t)$ values in every interval from 10^n to 10^{n+1} for $n = 16, 17, \ldots, 32$ respectively.

10^n	t	$Z(t)$	$\varphi(t)$
$n = 16$	54671407423795346.46	1081.06	0.181
$n = 17$	356071078353654562.22	1287.14	0.177
$n = 18$	5766261201333823098.71	−1634.103	0.171
$n = 19$	14497714038556679490.57	1527.573	0.166
$n = 20$	725415638078567550742.50	−2188.56	0.160
$n = 21$	6436526919750171929565.996	−2944.20	0.159
$n = 22$	2591147898973822635113 1.25	2748.79	0.153
$n = 23$	1262178822497143863467 58.12	3107.12	0.151
$n = 24$	1378580047424597442940453.455	3615.45	0.147
$n = 25$	7302710921631554712597461 5.93	5303.99	0.144
$n = 26$	8176298389276747155871879 88.46	−5381.06	0.139
$n = 27$	2973386013776701863168636829.09	6923.711	0.140
$n = 28$	3289101534777837101421589 5196.49	7268.43	0.135
$n = 29$	4908169772369533489867618 37011.71	−9971.49	0.135
$n = 30$	1270978758060014023140181770834.24	10905.332	0.134
$n = 31$	2845670144939668837484722465 5196.74	−15484.420	0.133
$n = 32$	3106788336290839656675405765 93682.05	16874.202	0.130

TABLE 5.1: Largest $\varphi(t)$ values found by the Riemann Zeta Search Project

The Lindelöf hypothesis is a conjecture about the rate of growth of the Riemann zeta function on the critical line that is implied by the Riemann hypothesis. It states that

for any $\epsilon > 0$

$$\zeta\left(\frac{1}{2} + it\right) = \mathcal{O}(t^{\epsilon})\,. \tag{5.1}$$

In Figure 5.1 one can see the descending tendency of $\varphi(t)$. As n goes to infinity, $\varphi(t)$ gets smaller and smaller. The red line is the actual best known theoretical bound for φ, proven by Bourgain (see 4.2).

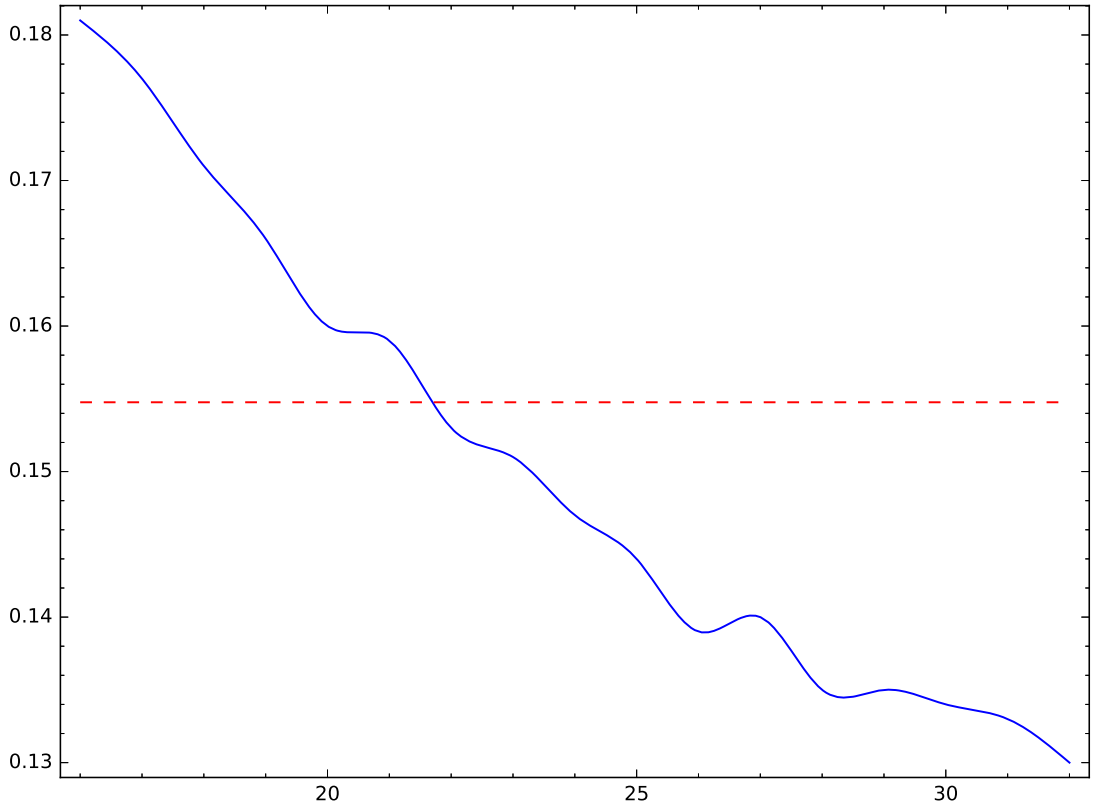

FIGURE 5.1: The descending $\varphi(t)$ numerically supports the Lindelöf hypothesis.

Based on numerical evidence and calculations (see Table 5.1) coming from the Riemann Zeta Search Project one can observe that the descending values of $\varphi(t)$ numerically support the Lindelöf hypothesis.

We have computed thousands of large $Z(t)$ values and this huge amount of data can be used to search for new patterns in the behaviour of the Riemann zeta function. The next step is to find as many large Z(t) values as possible with very large $\varphi(t)$ to get a better understanding of the behaviour of the zeta function near such unusually large values.

www.riemann-siegel.com

The author of this PhD thesis created a project website for the Riemann Zeta Search Project. Every achievement is published on www.riemann-sigel.com. Publications, new records and thousands of large Z(t) values can be downloaded from the project website.

Further research

It has turned out that $F(t)$ can be used to generate random numbers very efficiently. In order to investigate the quality of the random numbers generated by $F(t)$ we used the National Institute of Standards and Technology (NIST) randomness test suite.

The NIST test suite is a statistical package consisting of 15 statistical tests that were developed to test the randomness of arbitrarily long binary sequences produced by either hardware or software based cryptographic random or pseudorandom number generators. In case of each statistical test a set of P-values was produced.

Given a significance level α, if the P-value is less than or equal to α then the test suggests that the observed data is inconsistent with our null hypothesis, i.e. the 'hypothesis of randomness', so we reject it. We used $\alpha = 0.01$ to test the output of the $F(t)$ function. An α of 0.01 indicates that one would expect 1 sequence in 100 sequences to be rejected under the null hypothesis. Hence a P-value exceeding 0.01 would mean that the sequence would be considered to be random, and P-value ≤ 0.01 would lead to the conclusion that the sequence is non-random.

The uniformity of P-values produced by $F(t)$ is evenly distributed and it can be concluded that $F(t)$ produced very high quality random numbers. This is a very interesting connection between the zeta function and cryptography. We would like to continue our research in this direction.

Summary

In this PhD thesis we presented an efficient algorithm called `RS-PEAK` which can be used to find extremely large values of the Riemann zeta function on the critical line. Locating peak values of the zeta function is a promising method for getting a better understanding of the distribution of prime numbers.

In Chapter 2 we investigated multidimensional approximation problems. Approximating $\cos(\theta(t) - \frac{2k\pi}{\log 2} \cdot \log n)$ well with $\cos(\theta(t))$ for as many n as possible is a simultaneous Diophantine approximation problem. We created an efficient algorithm called `MAFRA` to solve n-dimensional Diophantine approximations. `MAFRA` was used to generate the candidates k where large $Z(\frac{2k\pi}{\log 2})$ values were expected. Weak candidates were then eliminated by a special function

$$F(t) = \sum_{n=1}^{\lfloor \log t/2\pi \rfloor} \frac{1}{\sqrt{n}} \cos(\theta(t) - t \cdot \log n)\,.$$

Regarding the peak values it has turned out that in most cases $Z(t)$ and $F(t)$ have the same behaviour. Note that the complexity of $F(t)$ is $\mathcal{O}(\log t)$, which is significantly better than the computational complexity of $Z(t)$.

Using `MAFRA` and relying on the special behaviour of $F(t)$ we created the `RS-PEAK` algorithm by which we were able to locate many large values of the Riemann zeta function on the critical line. The largest $Z(t)$ value found by the `RS-PEAK` algorithm is

$$Z(3106788336290839656675405765936 82.05) \approx 16874.202$$

To the best of our knowledge, at the time of writing of this PhD thesis this is the largest $Z(t)$ ever calculated. This value was published and presented in the Journal of Grid Computing [55] in 2017 as a new record. The value was verified by the ATLAS Super computing cluster operating at Eötvös Loránd University, Hungary.

Summary in Hungarian - Magyar összefoglaló

PhD értekezésemben egy olyan hatékony algoritmust mutattam be, amely a Riemann-Siegel Z-függvény kiugró értékeinek meghatározására szolgál. A Riemann-féle zeta függvény nagyon fontos szerepet játszik a matematika és a fizika különböző területein. A zeta függvény kritikus egyenesen elhelyezkedő nagy értékeinek meghatározása hozzásegíthet minket a prímszámok eloszlásának sokkal jobb megértéséhez.

A 2. fejezetben egy olyan algoritmust vizsgáltunk, amelynek segítségével gyorsan és hatékonyan tudtuk közelíteni a $\cos\left(\theta(t) - \frac{2k\pi}{\log 2} \cdot \log n\right)$-t nagyon sok n egészre.

Módszerünk többdimenziós szimultán Diofantikus egyenletek approximációján alapul, melynek megoldására hatékony algoritmust mutattunk be (`MAFRA` algoritmus).

Ezt felhasználva, illetve az

$$F(t) = \sum_{n=1}^{\lfloor \log t/2\pi \rfloor} \frac{1}{\sqrt{n}} \cos\left(\theta(t) - t \cdot \log n\right)$$

függvényt bevezetve (3. fejezet) hatékony algoritmust adtunk (`RS-PEAK` algoritmus) kiugróan magas $Z(t)$ értékek lokalizálására. A fenti $F(t)$ függvény nagyon hasonlóan viselkedik, mint az eredeti $Z(t)$ függvény, de kiszámítása csak $\mathcal{O}(\log t)$ időt vesz igénybe. Megállapítottuk, hogy $F(t)$ kiugró értékei sok esetben ott következnek be ahol $Z(t)$-nek is kiugró értékei várhatóak.

A 4. fejezetben az `RS-PEAK` algoritmus alkalmazásával számos nagy $Z(t)$ értékeket mutattunk be. Az algoritmus segítségével az MTA SZTAKI Desktop GRID hálózatát felhasználva sikerült nagyon nagy $Z(t)$ értékeket publikálni [55], köztük a ma ismert legnagyobbat is, ahol $t = 3106788336290839656675405765936 82.05$-ra $Z(t) \approx 16874.202$ értéket kapjuk. **Legjobb tudásunk szerint a disszertáció írásának idején ez a legnagyobb ismert $Z(t)$ érték, amit valaha kiszámoltak.**

Appendix A

Algorithms

A.1 FindMM Algorithm

Description:

The algorithm is based on Theorem 2.5. It finds the smallest integers m_1, m_2 and m_3 such that $0 < \langle m_1 \rangle < 2\varepsilon$, $-2\varepsilon < \langle m_2 \rangle < 0$. The output of the algorithm is $\Delta_\Omega = \{m_1, m_2, m_1 + m_2\}$. The main **while** loop in this algorithm (from line 5 to 15) goes through all intermediate convergents to find m_1 and m_2. When m_1 and m_2 are found the **while** loop terminates and the algorithm returns m_1,m_2 and $m_1 + m_2$ in ascending order.

Algorithm 2 FindMMM

Precondition: $\alpha \in \mathbb{R} \setminus \mathbb{Q}, \alpha > \varepsilon > 0$.

```
 1: procedure FindMMM(α, ε)
 2:     i ← 0
 3:     m_1 ← 0
 4:     m_2 ← 0
 5:     while m_1 = 0 or m_2 = 0 do
 6:         i ← i + 1
 7:         q_i ← i^th intermediate convergents of α
 8:         k ← Frac(q_i · α)                    ▷ Fractional part of q_i · α
 9:         if m_1 = 0 and k < 2ε then
10:             m_1 ← q_i
11:         end if
12:         if m_2 = 0 and k > 1 − 2ε then
13:             m_2 ← q_i
14:         end if
15:     end while
16:     Return(min(m_1, m_2), max(m_1, m_2), m_1 + m_2)
17: end procedure
```

A.2 Challenge 1 Solver Algorithm

Description:

The algorithm solves Challenge (2.12). Line 5 calls the FindMMM algorithm to determine Δ_Ω. With the theory of continued fractions line 6 finds an integer $k \in \Omega$. In the first **while** loop (lines 9–18) the appropriate m_i is subtracted from k to generate a new integer $k_i \in \Omega$. The process is repeated until the lower bound A of the interval is reached. In the second **while** loop (lines 20–29) the appropriate m_i is added to k generating $k_i \in \Omega$. The process is repeated until the upper bound of the interval B is reached. This method produces all the 18 000 integers that satisfy Challenge 1 (see 2.12).

Algorithm 3 Challenge 1 Solver

1: $x \leftarrow \sqrt{2}$
2: $\varepsilon \leftarrow 10^{-17}$
3: $A \leftarrow 10^{20}$
4: $B \leftarrow 10^{21}$
5: $v \leftarrow FindMMM(x, \varepsilon)$
6: $k \leftarrow$ Find q_x in the interval [A,B] where $\text{FRAC}(q_x \cdot x) < \varepsilon$
7: $ktemp \leftarrow k$
8: $\text{PRINT}(k)$
9: **while** $k > A$ **do**
10: **for** $i = 1 \rightarrow 3$ **do**
11: $ok \leftarrow \text{FRAC}((k - v[i]) \cdot x)$
12: **if** $(ok < \varepsilon)$ or $(ok > 1 - \varepsilon)$ **then** $k \leftarrow k - v[i]$
13: **if** $k > A$ **then** $\text{PRINT}(k)$
14: **end if**
15: **break** ▷ Leave the **for** loop
16: **end if**
17: **end for**
18: **end while**
19: $k \leftarrow ktemp$
20: **while** $k < B$ **do**
21: **for** $i = 1 \rightarrow 3$ **do**
22: $ok \leftarrow \text{FRAC}((k + v[i]) \cdot x)$
23: **if** $(ok < \varepsilon)$ or $(ok > 1 - \varepsilon)$ **then** $k \leftarrow k + v[i]$
24: **if** $k < B$ **then** $\text{PRINT}(k)$
25: **end if**
26: **break** ▷ Leave the **for** loop
27: **end if**
28: **end for**
29: **end while**

A.3 Challenge 2 Solver Algorithm

Description:

The algorithm solves Challenge 2.13. Line 5 calls the PRECALC algorithm in order to determine the 2^n integers. The **while** loop generates a new integer in Ω using the precalculated ones. The method produces 120 852 integers that satisfy Challenge 2.13.

Algorithm 4 Challange 2 Solver

1: $n \leftarrow 7$
2: $X \leftarrow \frac{\log(p)}{\log(2)}, p \text{ prime}, 3 \leq p \leq 19$
3: $\varepsilon \leftarrow 0.01$
4: $B \leftarrow 10^{18}$
5: $v \leftarrow \text{PRECALC}(n, \varepsilon, X, 2^{12})$
6: $k \leftarrow 0$
7: **while** $k < B$ **do**
8: **for** $i = 1 \rightarrow \text{length}(v)$ **do**
9: $t \leftarrow \text{TRUE}$
10: **for** $j = 1 \rightarrow n$ **do**
11: $ok \leftarrow \text{FRAC}((k + v[i]) \cdot X[j])$
12: **if** $(ok > \varepsilon)$ and $(ok < 1 - \varepsilon)$ **then**
13: $t \leftarrow \text{FALSE}$
14: **break** ▷ Leave the **for** loop
15: **end if**
16: **end for**
17: **if** $t = \text{TRUE}$ **then**
18: $k \leftarrow k + v[i]$
19: **if** $k < B$ **then**
20: $\text{PRINT}(k)$
21: **end if**
22: **break**
23: **end if**
24: **end for**
25: **end while**

A.4 Reduce Algorithm

Description:

The algorithm reduces the generation time of Γ_n in the **Precalc** algorithm with adding new elements to the integer list K. X is a set of irrationals such that $\|K[i] \cdot X[j]\| < \varepsilon$ for all i and for all $j < n$. The main part of the algorithm is the for loop (lines 4–9). Each element of K is subtracted (added) from (to) every element of K and the new integer k_i that satisfies $\|k_i \cdot X[j]\| < \varepsilon$ for all $j < n$ are appended to K.

Algorithm 5 Reduce

Precondition: K: list of integers, $n \in \mathbb{N}$, $\varepsilon > 0$, X: set of irrationals

1: **procedure** REDUCE(K, n, ε, X)
2: SORT(K) ▷ Sorting, every element ouccurs only once
3: $M \leftarrow$ dynamic array()
4: **for** $i = 1 \rightarrow$ length(K) **do**
5: **for** $j = 1 \rightarrow$ length(K) **do**
6: APPEND(M, abs($K[i] - K[j]$)) ▷ append abs($K[i] - K[j]$) to M
7: APPEND(M, abs($K[i] + K[j]$))
8: **end for**
9: **end for**
10: SORT(M)
11: **for** $i = 1 \rightarrow$ length(M) **do**
12: $t \leftarrow$ TRUE
13: **for** $j = 1 \rightarrow n$ **do**
14: $t \leftarrow t$ **and** (FRAC($M[i] \cdot X[j]$) $< 2\varepsilon$ **or** FRAC($M[i] \cdot X[j]$) $> 1 - 2\varepsilon$)
15: **end for**
16: **if** $t =$ FALSE **then**
17: DELETE($M[i]$) ▷ Delete the i^{th} element of M
18: **end if**
19: **end for**
20: APPEND(K, M) ▷ Append array M to K
21: SORT(K)
22: **if** $K[1] = 0$ **then**
23: DELETE($K[1]$) ▷ Delete the zero value from K
24: **end if**
25: RETURN(K)
26: **end procedure**

A.5 Precalc Algorithm

Description:

The algorithm is based on **Theorem 2.6**. It generates Γ_n, a subset of Δ_Ω. In dimension n the set Γ_n contains exactly 2^n elements. Initially, the **FindMMM** algorithm is invoked in line 2. In higher dimensions $(2, 3, \ldots$ up to $m)$ the algorithm produces many integers from Δ_Ω by which Γ_n can be generated. M is a matrix with i rows. The i^{th} row contains the binary representation of i. (Note: the size of M is changing depending on the dimension.) To produce as many integers as possible the **Reduce** algorithm is used (see lines $10, 11$). If β goes to infinity then Δ_Ω should contain almost all possible step-sizes.

For example in order to solve Challenge 2.13 we set $\beta = 2^{12}$. With this choice of β the algorithm is able to generate the appropriate Γ_n up to dimension 10. For higher dimensions bigger β is needed.

Algorithm 6 Precalc

```
1:  procedure PRECALC(m, ε, X, β)
2:      T ← FINDMMM(X[1], ε)                          ▷ T is a dynamic array
3:      for n = 2 → m do
4:          T2 ← dynamic array()
5:          N ← 0, T3 ← 0        ▷ N, T3 are arrays with 2^n elements, every element is 0
6:          M ← 2^n × n matrix, the i^th row contains the binary representation of i
7:          k ← 0, tmp ← 0, l ← 0, number ← 0
8:          while TRUE do
9:              if l = 2^n and number > β then
10:                 REDUCE(T2, n, ε, X)
11:                 REDUCE(T2, n, ε, X)
12:                 T ← T2
13:                 break                              ▷ Leave the while loop
14:             end if
15:             for i = 1 → length(T) do
16:                 t ← TRUE
17:                 for j = 1 → n − 1 do
18:                     ok ← FRAC((k + T[i]) · X[j])
19:                     if ok > ε and ok < 1 − ε then
20:                         t ← FALSE
21:                         break                      ▷ Leave the for loop
22:                     end if
23:                     if t = TRUE then
24:                         k ← k + T[i]
25:                         break                      ▷ Leave the for loop
26:                     end if
27:                 end for
28:             end for
```

Algorithm 7 Precalc (contd.)

```
29:             number ← number + 1
30:             t ← TRUE
31:             for j = 1 → n do
32:                 t ← t and (FRAC(k · X[j]) < ε or FRAC(k · X[j]) > 1 − ε)
33:             end for
34:             if t = FALSE then
35:                 next
36:             end if
37:             t ← FALSE
38:             for i = 1 → length(T2) do
39:                 if T2[i] = k − tmp then
40:                     t ← TRUE
41:                 end if
42:             end for
43:             if t = FALSE then
44:                 APPEND(T2, k − tmp)          ▷ append k − tmp to the array T2
45:             end if
46:             tmp ← k
47:             for i = 1 → 2^n do
48:                 t ← TRUE
49:                 for j = 1 → n do
50:                     if M[i, j] = 0 then
51:                         t ← t and (FRAC(k · X[j]) < ε)
52:                     else
53:                         t ← t and (FRAC(k · X[j]) > 1 − ε)
54:                     end if
55:                 end for
56:                 if t and N[i] = 0 then
57:                     N[i] ← 1
58:                     l ← l + 1
59:                     T3[l] ← k
60:                     if l = 2^n and n = m then
61:                         break(2)                 ▷ Leave while loop
62:                     end if
63:                 end if
64:             end for
65:         end while
66:     end for
67:     RETURN(T3)
68: end procedure
```

A.6 Fast Rational Approximation (FRA) Algorithm

Algorithm 8 – (FRA) – Fast Rational Approximation

Precondition: `bound` ▷ default is one billion
Precondition: k ▷ starting point, the default is zero

$\Gamma \leftarrow \textsc{Precalc}(n, \varepsilon, \Upsilon, 2^{12})$
$\Upsilon \leftarrow \frac{\log(p)}{\log(2)},\ p$ prime, $3 \leq p \leq 31$
`counter` $\leftarrow 0$, $\varepsilon \leftarrow 0.01$
while `counter` $<$ `bound` **do**
    for $i = 1 \rightarrow 1024$ **do**
        `find` $\leftarrow$ TRUE
        for $j = 1 \rightarrow 10$ **do**
            $a \leftarrow \textsc{Frac}((k + \Gamma[i]) \cdot \Upsilon[j])$
            if $(a > \varepsilon)$ and $(a < 1 - \varepsilon)$ **then**
                `find` $\leftarrow$ FALSE
                break ▷ Leave the **for** loop
            end if
        end for
        if `find` $=$ TRUE **then**
            $k \leftarrow k + \Gamma[i]$
            `counter` $\leftarrow$ `counter` $+ 1$
            break
        end if
    end for
end while

A.7 Advanced Fast Rational Approximation (AFRA) Algorithm

Algorithm 9 – (AFRA) – Advanced Fast Rational Approximation

Precondition: `bound`
Precondition: k ▷ starting point, the default is zero

1: $\Gamma \leftarrow \textsc{Precalc}(n, \varepsilon, \Upsilon, 2^{12})$
2: $\Upsilon \leftarrow \frac{\log(p)}{\log(2)},\ p$ prime, $3 \leq p \leq 31$
3: $\varepsilon \leftarrow 0.01$
4: `counter` $\leftarrow 0$
5: **while** `counter` $<$ `bound` **do**
6: `sum` $\leftarrow 0$
7: **for** $i = 1 \rightarrow 10$ **do**
8: $a \leftarrow \textsc{Frac}(k \cdot \Upsilon[i])$
9: **if** $(a < \varepsilon)$ **then**
10: `sum` $\leftarrow$ `sum` $+ 2^i$
11: **end if**
12: **end for**
13: `sum` $\leftarrow abs($`sum` $- 1024)$ ▷ binary complementer
14: `counter` $\leftarrow$ `counter` $+ 1$
15: $k \leftarrow k + \Gamma[$`sum`$]$
16: **end while**

A.8 RS-PEAK Algorithm

Algorithm 10 RS-Peak

procedure RS-PEAK($n, \varepsilon, C_1, C_2, C_3, C_4, C_5, B_1, B_2$)
  $\Upsilon \leftarrow \frac{\log(p)}{\log(2)},\ p$ prime, $3 \le p \le 31$ ▷ Array of irrationals
  $\varepsilon \leftarrow 0.01, \quad \Gamma \leftarrow$ PRECALC$(n, \varepsilon, \Upsilon, 2^{12})$
  $k \leftarrow k_1$ from $\Omega(\Upsilon, \varepsilon, B_1, B_2)$ ▷ First integer from Ω
  while $k < B_2$ **do**
    for $i = 1 \to 2^n$ **do**
      $t \leftarrow$ TRUE
      for $j = 1 \to n$ **do**
        $ok \leftarrow$ FRAC$((k + \Gamma[i]) \cdot \Upsilon[j])$
        if $(ok > \varepsilon)$ **and** $(ok < 1 - \varepsilon)$ **then**
          $t \leftarrow$ FALSE
          break ▷ Leave the **for** loop
        end if
      end for
      if $t =$ TRUE **then**
        $k \leftarrow k + \Gamma[i]$
        $t \leftarrow \frac{2k\pi}{\log 2}$
        $a \leftarrow \texttt{abs}(\texttt{A}(t, 1, 1))$
        if $a > C_1$ **then**
          $a \leftarrow \texttt{abs}(\texttt{F}(t))$
          if $a > C_2$ **then**
            $a \leftarrow \texttt{abs}(\texttt{A}(t, 1, 100))$
            if $a > C_3$ **then**
              $a \leftarrow \texttt{abs}(\texttt{A}(t, 1, 1000))$
              if $a > C_4$ **then**
                $a \leftarrow \texttt{abs}(\texttt{A}(t, 10^6, 1.1 \cdot 10^6))$
                if $a > C_5$ **then**
                  if $k < B_2$ **then**
                    `print(t)` ▷ Large $Z(t)$ is expected
                  end if
                end if
              end if
            end if
          end if
        end if
        break
      end if
    end for
  end while
end procedure

Bibliography

[1] M. V. Berry and J. P. Keating *The Riemann zeros and eigenvalue asymptotics*, SIAM Review, 41 (1999) No. 2, 236–266.

[2] N. Shukla, *Construction of Infinitely Many Models of the Universe on the Riemann Hypothesis*, Reports on Mathematical Physics, Volume 78, Issue 2, (2016), 253–258.

[3] GIMPS Project, *The largest known prime number* $2^{82589933} - 1$, `https://www.mersenne.org/primes/?press=M82589933`, Last visited: 2019 April.

[4] J. Hadamard, *Sur la distribution des zéros de la fonction* $\zeta(s)$ *et ses conséquences arithmétiques*, Bull. Soc. Math. France 24 (1896), 199—220.

[5] C.J. de la Vallée Poussin, *Recherches analytiques sur la théorie des nombres premiers*, Ann. Soc Sci. Bruxelles 20 (1896), 183—256.

[6] N.M. Korobov, *Estimates for trigonometric sums and their applications*, Uspehi Mat. Nauk 13 (1958), 185–192.

[7] I.M. Vinogradov, *A new estimate for* $\zeta(1 + it)$, Izv. Akad. Nauk SSSR, Ser. Mat. 22 (1958), 161–164.

[8] D.A. Goldston, J. Pintz, C.Y. Yíldírím, *Primes in tuples I*, Annals of Math., Volume 170, Issue 2, (2009), 819–862.

[9] Y. Zhang, *Bounded gaps between primes*, Annals of Math., Volume 179, Issue 3, (2014), 1121–1174.

[10] Maynard, James, *Small gaps between primes*, Annals of Math., Volume 181, Issue 1, (2015), 383–413.

[11] L. Schoenfeld, *Sharper bounds for the Chebyshev functions* $\theta(x)$ *and* $\psi(x)$ *II.*, Math. Comp 30, (1976), 337–360.

[12] G.B.F. Riemann, *Über die Anzahl der Primzahlen unter einer gegebenen Grösse*, Monatsber. Königl. Preuss. Akad. Wiss. Berlin (1859), 671–680.

[13] R.P. Brent, *On the Zeros of the Riemann Zeta Function in the Critical Strip*, Math. Comp., Volume 33, Number 148, (1979), 1361–1372.

[14] J. van de Lune, H.J.J te Riele and D.T Winter, *On the Zeros of the Riemann Zeta Function in the Critical Strip. IV*, Math. Comp., Volume 46, Number 174, (1986), 667–681.

[15] A.M. Odlyzko, A. Schönhage, *Fast algorithms for multiple evaluations of the Riemann zeta function*, Trans. Am. Math. Soc., 309, (1988), 797–809.

[16] X. Gourdon, *The* 10^{13}*-rst zeros of the Riemann Zeta function, and zeros computation at very large height*,
`http://numbers.computation.free.fr/Constants/Miscellaneous/zetazeros1e13-1e24.pdf`, Last visited: 2018 september

[17] J.E. Littlewood, *Quelques conséquences de l'hypothèse que la fonction* $\zeta(s)$ *n'a pas de zéros dans le demi-plan* $\mathrm{Re}(s) > \frac{1}{2}$. C. R. Acad. Sci. Paris, (1912), 154, 263–266.

[18] A.M. Odlyzko , H.J.J. te Riele, *Disproof of the Mertens conjecture*, J. Reine Angew. Math. (1985), 357, 138–160.

[19] J. Pintz , *An effective disproof of the Mertens conjecture*. Astérisque. (1987), 147-148, 325–333 and 346.

[20] T. Kotnik, H.J.J. te Riele, *Mertens conjecture revisited*, Lecture Notes in Comput. Sci. 4076, (2006), 156–167.

[21] Gy. Pólya, *Verschiedene bemerkungen zur zahlentheorie*, Jahresbericht der Deutschen Mathematiker-Vereinigung, (1919), 28, 31–40.

[22] C.B. Haselgrove, *A disproof of a conjecture of Pólya*, Mathematika. (1958), 5 (02), 141–145.

[23] R.S. Lehman, *On Liouville's function*, Mathematics of Computation (1960), 14 (72), 311—320.

[24] M. Tanaka, *A Numerical Investigation on Cumulative Sum of the Liouville Function*, Tokyo Journal of Mathematics (1980), 3 (1), 187—189.

[25] H. von Koch, *Sur la distribution des nombres premiers*, Acta Math., vol. 24, (1901), 159–182.

[26] H. von Mangoldt, *Zu Riemanns Abhandlung "Über die Anzahl der Primzahlen unter einer gegebenen Grösse"*, Journal für die reine und angewandte Mathematik, vol. 114 (1895), 255–305.

[27] G.H. Hardy, *Sur les zeros de la fonction* $\zeta(s)$, Compt. Rend. Acad. Sci., (1914), 158, 1012–1014.

[28] G.H. Hardy, J.E. Littlewood, *The zeros of Riemann's Zeta-Function on the critical line* , Math. Zeit., 10 (1921), 283–317.

[29] A. Selberg, *On the zeros of Riemann's zeta-function*, Skr. Norske Vid. Akad. Oslo. 10, (1942), 1–59.

[30] N. Levinson, *More than one third of zeros of Riemann's zeta-function are on* $\sigma = 1/2$, Advances in Mathematics. Vol 13, Issue 4, (1974), 383–436.

[31] J.B. Conrey, *At least two fifths of the zeros of the Riemann zeta function are on the critical line*, Bull. Amer. Math. Soc. (N.S), Volume 20, Number 1 (1989), 79–81.

[32] E.C. Titchmarsh, *The theory of the Riemann zeta-function*, Clarendon Press Oxford, (1986).

[33] R.S. Lehman, *Separation of Zeros of the Riemann Zeta-Function*, Math. Comp., Volume 20, (1966), 523–541.

[34] G.A. Hiary, *Fast methods to compute the Riemann zeta function*, Ann. Math., 174-2, (2011), 891–946.

[35] T. Trudgian, *On the success and failure of Gram's Law and the Rosser Rule*, Acta Arithmetica 148 (2011), 225–256.

[36] K. Broughan, *Equivalents of the Riemann Hypothesis, Volume Two: Analytic Equivalents*, Cambridge University Press, (2017) ISBN:9781108178228, p. 79, Theorem 5.13.

[37] A.M. Odlyzko, *The* 10^{20}*-th zero of the Riemann zeta function and* 175 *million of its neighbors*, (1992), Webpage: `http://www.dtc.umn.edu/odlyzko/unpublished/`, Last visited: September 2018.

[38] T. Kotnik, *Computational estimation of the order of* $\zeta(1/2 + it)$, Math. Comp., Volume 73, Number 246, (2004), 949–956.

[39] J.W. Bober, G.A. Hiary *New computations of the Riemann zeta function on the critical line*, Experimental Mathematics (2016), 1–13.

[40] A.K. Lenstra, H.W. Lenstra Jr., L. Lovász , *Factoring polynomials with rational coefficients*, Math. Ann., 261, (4), (1982), 515–534.

[41] F. Armknecht, C. Elsner, M. Schmidt, *Using the Inhomogeneous Simultaneous Approximation Problem for Cryptographic Design*, AFRICACRYPT, 2011, 242–259.

[42] A. Frank, É. Tardos, *An application of simultaneous Diophantine approximation in combinatorial optimization*, Combinatorica, 7, 1 (1987) 49–66.

[43] H. Davenport's, *The Higher Arithmetic: An Introduction to the Theory of Numbers* (2008) , Cambridge University Press , ISBN: 9780511818097

[44] A.Y. Khinchin, *Continued Fractions, Translated from the third , Russian edition, Reprint of the 1964 translation* (1961) , Dover, Mineola, NY, 1997.

[45] Sh. Kim, S. Östlund, *Simultaneous rational approximations in the study of dynamical systems*, Phys. Rev. A, 34, 4 (1986) 3426–3434.

[46] C. Kimberling, *Best lower and upper approximates to irrational numbers*, Elem. Math., 52, 3 (1997) 122–126.

[47] J.C. Lagarias, *Best simultaneous Diophantine approximations I., Growth rates of best approximation denominators*, Trans. Am. Math. Soc., 272, 2 (1982) 545–554.

[48] J.C. Lagarias, *Best simultaneous Diophantine approximations II., Behavior of consecutive best approximations*, Pacific J. Math., 102, 1 (1982) 61–88.

[49] J.C. Lagarias, *The computational complexity of simultaneous Diophantine approximation problems*, SIAM J. Computing, 14, 1 (1985) 196–209.

[50] V. T. Sós, G. Szekeres, *Rational approximation vectors*, Acta Arithm., 49, 3 (1988) 255–261.

[51] V. T. Sós, *On the theory of diophantine approximation*s., Acta Math. Acad. Sci. Hungar. 8 (1957) 461–472.

[52] Świerckowski S., *On successive settings of an arc on the circumference of a circle.*, Fund. Math. 48 (1958), 187–189.

[53] J. Surányi, *Über der Anordnung der Vielfachen einer reelen Zahl mod 1* Ann. Univ. Sci. Budapest. Eötvös Sect. Math. 1 (1958), 107–111.

[54] A. Thall, *Extended-Precision Floating-Point Numbers for GPU Computation* , SIGGRAPH '06 ACM SIGGRAPH (2006) Research poster.

[55] N. Tihanyi, A. Kovács, J. Kovács, *Computing Extremely Large Values of the Riemann Zeta Function*, J. Grid Computing (2017) 15, 527–534. https://doi.org/10.1007/s10723-017-9416-0

[56] N. Tihanyi, *Fast Method for Locating Peak Values of the Riemann Zeta Function on the Critical Line*, 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2014 , IEEE Explorer 10.1109/SYNASC.2014.20

[57] A. Kovács, N. Tihanyi, *Efficient computing of n-dimensional simultaneous Diophantie approximation problems*, Acta Univ. Sapientiae, Informatica 5-1, (2013), 16–34.

[58] N. Tihanyi, A. Kovács, Á. Szűcs, *Distributed computing of simultaneous Diophantine approximation problems*, Stud. Univ. Babes-Bolyai Math. 59, (2014), No. 4, 557–566.

[59] N. Tihanyi, A. Kovács, Á. Szűcs Distributed computing of n-dimensional simultaneous Diophantine approximation problems, 10th Joint Conf. on Math. and Comp. Sci, (2014) , Cluj-Napoca, Romania, May 22–25.

[60] N. Tihanyi, A. Kovács, Improvements on finding large candidates of the Riemann zeta function on the critical line, Annales Univ. Sci. Budapest., Sect. Comp. 48 (2018), pp.53–64

[61] N. Tihanyi, A. Kovács, *Computation Results of the Riemann Zeta Search Project*, 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2018, (accepted).

[62] A.M. Odlyzko, H.J.J te Riele, *Disproof of the Mertens conjecture*, Journal für die reine und angewandte Mathematik, 357, 138—160, (1985).

[63] G.A. Hiary, *A nearly-optimal method to compute the truncated theta function, its derivatives, and integrals*, Annals of Math., Volume 174, (2011), 859–889.

[64] Jean Bourgain, *Decoupling, exponential sums and the Riemann zeta function*, Journal of the American Mathematical Society, Volume 30, Number 1 (2017), 205–224.

[65] David P. Anderson, *BOINC: A System for Public-Resource Computing and Storage.* In Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing (GRID '04)., IEEE Computer Society, Washington, DC, USA.

[66] P. Kacsuk, J. Kovács, Z. Farkas, *SZTAKI Desktop Grid (SZDG): A Flexible and Scalable Desktop Grid System*, Journal of Grid Computing, Special Issue: Volunteer Computing and Desktop Grids, Vol. 7, Issue: 4, pp. 439–461, 2009.

[1]**ADATLAP**
a doktori értekezés nyilvánosságra hozatalához

I. A doktori értekezés adatai
A szerző neve: **Tihanyi Norbert**
MTMT-azonosító: **10033910**
A doktori értekezés címe és alcíme: **Numerical computing of extremely large values of the Riemann-Siegel Z-function**
DOI-azonosító[2]: **10.15476/ELTE.2019.089**
A doktori iskola neve: **ELTE Informatika Doktori Iskola**
A doktori iskolán belüli doktori program neve: **Numerikus és Szimbolikus Számítások**
A témavezető neve és tudományos fokozata: **Dr. Habil. Kovács Attila**
A témavezető munkahelye: **ELTE IK Komputeralgebra tanszék**

II. Nyilatkozatok
1. A doktori értekezés szerzőjeként[3]

a) hozzájárulok, hogy a doktori fokozat megszerzését követően a doktori értekezésem és a tézisek nyilvánosságra kerüljenek az ELTE Digitális Intézményi Tudástárban. Felhatalmazom az Informatika Doktori Iskola hivatalának ügyintézőjét, Kulcsár Adinát, hogy az értekezést és a téziseket feltöltse az ELTE Digitális Intézményi Tudástárba, és ennek során kitöltse a feltöltéshez szükséges nyilatkozatokat.

b) kérem, hogy a mellékelt kérelemben részletezett szabadalmi, illetőleg oltalmi bejelentés közzétételéig a doktori értekezést ne bocsássák nyilvánosságra az Egyetemi Könyvtárban és az ELTE Digitális Intézményi Tudástárban;[4]

c) kérem, hogy a nemzetbiztonsági okból minősített adatot tartalmazó doktori értekezést a minősítés (*dátum*)-ig tartó időtartama alatt ne bocsássák nyilvánosságra az Egyetemi Könyvtárban és az ELTE Digitális Intézményi Tudástárban;[5]

d) kérem, hogy a mű kiadására vonatkozó mellékelt kiadó szerződésre tekintettel a doktori értekezést a könyv megjelenéséig ne bocsássák nyilvánosságra az Egyetemi Könyvtárban, és az ELTE Digitális Intézményi Tudástárban csak a könyv bibliográfiai adatait tegyék közzé. Ha a könyv a fokozatszerzést követőn egy évig nem jelenik meg, hozzájárulok, hogy a doktori értekezésem és a tézisek nyilvánosságra kerüljenek az Egyetemi Könyvtárban és az ELTE Digitális Intézményi Tudástárban.[6]

2. A doktori értekezés szerzőjeként kijelentem, hogy

a) az ELTE Digitális Intézményi Tudástárba feltöltendő doktori értekezés és a tézisek saját eredeti, önálló szellemi munkám és legjobb tudomásom szerint nem sértem vele senki szerzői jogait;

b) a doktori értekezés és a tézisek nyomtatott változatai és az elektronikus adathordozón benyújtott tartalmak (szöveg és ábrák) mindenben megegyeznek.

3. A doktori értekezés szerzőjeként hozzájárulok a doktori értekezés és a tézisek szövegének plágiumkereső adatbázisba helyezéséhez és plágiumellenőrző vizsgálatok lefuttatásához.

Kelt: BP, 2019. 05. 20

a doktori értekezés szerzőjének aláírása

---

[1] Beiktatta az Egyetemi Doktori Szabályzat módosításáról szóló CXXXIX/2014. (VI. 30.) Szen. sz. határozat. Hatályos: 2014. VII.1. napjától.
[2] A kari hivatal ügyintézője tölti ki.
[3] A megfelelő szöveg aláhúzandó.
[4] A doktori értekezés benyújtásával egyidejűleg be kell adni a tudományági doktori tanácshoz a szabadalmi, illetőleg oltalmi bejelentést tanúsító okiratot és a nyilvánosságra hozatal elhalasztása iránti kérelmet.
[5] A doktori értekezés benyújtásával egyidejűleg be kell nyújtani a minősített adatra vonatkozó közokiratot.
[6] A doktori értekezés benyújtásával egyidejűleg be kell nyújtani a mű kiadásáról szóló kiadói szerződést.